Codes on Graphs: Duality and MacWilliams Identities

Codes on Graphs: Duality and MacWilliams Identities

G. David Forney, Jr.*

Abstract

A conceptual framework involving partition functions of normal factor graphs is introduced, paralleling a similar recent development by Al-Bashabsheh and Mao. The partition functions of dual normal factor graphs are shown to be a Fourier transform pair, whether or not the graphs have cycles. The original normal graph duality theorem follows as a corollary.

Within this framework, MacWilliams identities are found for various local and global weight generating functions of general group or linear codes on graphs; this generalizes and provides a concise proof of the MacWilliams identity for linear time-invariant convolutional codes that was recently found by Gluesing-Luerssen and Schneider. Further MacWilliams identities are developed for terminated convolutional codes, particularly for tail-biting codes, similar to those studied recently by Bocharova, Hug, Johannesson and Kudryashov.

Keywords: codes on graphs, MacWilliams identities, normal factor graphs, partition functions.

1 Introduction

Linear codes defined by graphical models have become the central subject of modern coding theory. Moreover, the topic of "codes on graphs" has proved to have rich connections to such cognate fields as system theory and statistical physics.

Duality has long been a prominent theme in the study of linear codes. Duality theory often yields simple, powerful, and surprisingly general results. Some problems become much simpler in the dual domain.

A fundamental duality result in the field of "codes on graphs" is the normal graph duality theorem of [5]. By imposing certain natural "normal" degree restrictions on graphical models, which are in fact not at all restrictive, the normal graph duality theorem shows how a graphical model for a dual linear code $\mathcal{C}^\perp$ may be obtained by local dualization of a graphical model for a linear code $\mathcal{C}$, whether or not the graph has cycles.

At the same time as [5] (in an adjacent paper in the same special issue), the conceptual framework of "factor graphs" was introduced by Kschischang, Frey, and Loeliger [11] to unify various styles of graphical models such as Tanner graphs, Bayesian networks, Markov random fields, Kalman filtering, and so forth, and the various computational algorithms that have been developed independently in these various fields. Subsequently, the "normal" restriction has been adopted to some extent in the factor graph literature, particularly by Loeliger and his co-authors (see, *e.g.,* [12, 13]).

Mao and Kschischang [15] have developed a general duality theory for factor graphs, particularly by introducing "convolutional" factor graphs as duals to conventional multiplicative factor graphs. By specializing to normal graphs, they proved the normal graph duality theorem as a corollary.

*The author is with the Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA 02139 (email: forneyd@comcast.net).

Very recently, Al-Bashabsheh and Mao [1] have shown that the normal graph duality theorem and Valiant's "holographic" algorithms can both be united within a common framework, which they call "normal factor graphs and holographic transformations." We have collaborated intensively with the authors of that paper and with the Associate Editor of both (P. Vontobel) while writing this paper, and have tried to achieve as much commonality as possible in our presentations.

As in [1], we regard normal factor graphs as representations of realizations that involve external variables, internal variables and factors. All variables are vector spaces over a given finite field $\mathbb{F}$, and all factors are complex-valued. Each factor involves some subset of the variables, with the following "normal" degree restrictions: each external variable is involved in precisely one factor, and each internal variable is involved in precisely two factors. As shown in [5], *any* realization may be straightforwardly "normalized" by a simple replication procedure that does not affect the topology of the associated graphical model. In the corresponding normal factor graph (NFG), factors are then represented by vertices, internal variables by ordinary edges of degree 2, and external variables by edges of degree 1 ("half-edges," "dangling edges," "dongles").

We define the *partition function* of such a normal factor graph as the sum over all internal variables of the product of all factors. (In [1], this is called the "exterior function" of the NFG.) In particular, if all factors are indicator functions of local linear constraint codes over $\mathbb{F}$, then the partition function is (up to scale) the indicator function of a global linear code $\mathcal{C}$.

With [1], we believe that the conceptual framework of representing sums of products as partition functions of normal factor graphs is an important paradigm that, surprisingly, does not seem to have been discussed very explicitly in the previous factor graph literature. We therefore present this framework in some generality in Sections 2 and 3.

In Section 3, we prove a general normal factor graph duality theorem (also proved in [1]), which shows that the partition functions of a normal factor graph and its dual are a Fourier transform pair, up to scale. Specializing to indicator functions of linear codes, we obtain as a corollary a result that is equivalent to the normal graph duality theorem of [5].

In Sections 4 and 5, we present further applications that were stimulated by several recent results on MacWilliams identities for linear convolutional codes [9, 10, 2].

For linear block codes, MacWilliams identities are classical duality results that relate weight generating functions of linear codes and their duals [14]. It was shown more than thirty years ago by Shearer and McEliece [16] that there is no MacWilliams identity for the usual weight generating function (the free distance spectrum) of a convolutional code.

However, Gluesing-Luerssen and Schneider (GLS) have recently formulated [9] and proved [10] a MacWilliams identity that involves the Hamming weight adjacency matrix (HWAM) of a linear time-invariant convolutional code over a finite field and the HWAM of its dual code.

In Section 2, we show how weight generating functions of various types may be naturally represented as partition functions of normal factor graphs. Furthermore, for a convolutional code, it is natural to replace a weight generating function (WGF) by a weight adjacency matrix (WAM).

In Section 4, we then apply our normal factor graph duality results to prove an appropriate MacWilliams identity between a WGF (or WAM) of a local linear code and a WGF (or WAM) of its dual. We consider exact, complete, and Hamming weight generating functions. This gives a concise proof of the GLS result, and generalizes it to arbitrary group codes defined on graphs; *e.g.,* linear block codes defined on trellises, linear tail-biting codes, or trellis codes over abelian groups. A preliminary version of these results was presented in [6].

Bocharova, Hug, Johannesson, and Kudryashov [2] have recently proved a MacWilliams identity that holds for truncations of a convolutional code $\mathcal{C}$ and its orthogonal code $\mathcal{C}^\perp$. As the truncation length becomes large, they obtain approximations to the free distance spectra of $\mathcal{C}$ and $\mathcal{C}^\perp$.

In Section 5, we develop MacWilliams identities for distance distributions of various kinds of terminated convolutional codes, particularly tail-biting terminated codes. A preliminary version of these results was presented in [7]. These results effectively answer the original question posed by Shearer and McEliece [16].

2 Codes, Realizations and Graphical Models

In this section we review linear codes, realizations of codes, and their graphical models. We start with the development and notation of [5], but then transition to normal factor graphs rather than normal graphs. We develop the general framework of partition functions of normal factor graphs. Finally, we show how weight generating functions are naturally represented in this framework.

2.1 Linear codes, realizations and normal graphs

In this paper we will be concerned with linear codes over a finite field $\mathbb{F}$. Everything generalizes to group codes over finite abelian groups, but for simplicity we will restrict attention to linear codes.

A linear code $\mathcal{C}$ over $\mathbb{F}$ is defined over a discrete index set $\mathcal{I}_\mathcal{A}$ and a set $\{A_k : k \in \mathcal{I}_\mathcal{A}\}$ of *symbol alphabets* A_k, each a finite-dimensional vector space over $\mathbb{F}$, and thus finite. The code $\mathcal{C}$ is then a subspace of the Cartesian-product vector space $\mathcal{A} = \Pi_{k\in\mathcal{I}} A_k$, called the *symbol sequence space*.

In general, the index set $\mathcal{I}_\mathcal{A}$ may be infinite; however, in this paper we will assume for simplicity that $\mathcal{I}_\mathcal{A}$ is finite, so that the symbol sequence space $\mathcal{A}$ is finite. For convolutional codes, this assumption may be justified by considering an infinitely long convolutional code as a limit of a sequence of finitely long terminated codes; see Section 5.

A *realization* of a code $\mathcal{C}$ is a concrete implementation or characterization of it. For instance, a parity-check realization of a linear (n, k) code $\mathcal{C}$ characterizes it as the set of all $\mathbf{a} \in \mathbb{F}^n$ that satisfy a set of $n - k$ parity-check equations. A *Tanner graph* is a graphical model of such a realization.

More generally, a *behavioral realization* of $\mathcal{C}$ involves not only the set $\{A_k : k \in \mathcal{I}_\mathcal{A}\}$ of symbol alphabets, but also a set $\{S_j : j \in \mathcal{I}_\mathcal{S}\}$ of auxiliary alphabets, often called *state spaces*, indexed by a state index set $\mathcal{I}_\mathcal{S}$, and a set $\{\mathcal{C}_i : i \in \mathcal{I}_\mathcal{C}\}$ of local constraint codes $\mathcal{C}_i$ indexed by a constraint index set $\mathcal{I}_\mathcal{C}$, where each constraint code $\mathcal{C}_i$ involves some subsets $\mathbf{a}_i$ and $\mathbf{s}_i$ of the symbol and state variables, respectively. In a linear behavioral realization, each state space S_j and each constraint code $\mathcal{C}_i$ is a vector space over $\mathbb{F}$. We define the state sequence space as $\mathcal{S} = \Pi_{j\in\mathcal{I}_\mathcal{S}} S_j$.

The *full behavior* of the realization is the set $\mathfrak{B}$ of all pairs $(\mathbf{a}, \mathbf{s}) \in \mathcal{A}\times\mathcal{S}$ such that all constraints are satisfied; *i.e.,* $(\mathbf{a}_i, \mathbf{s}_i) \in \mathcal{C}_i, \forall i \in \mathcal{I}_\mathcal{C}$. The *code* $\mathcal{C}$ generated by the realization is then the set of all symbol sequences $\mathbf{a} \in \mathcal{A}$ that appear in some $(\mathbf{a}, \mathbf{s}) \in \mathfrak{B}$.

For example, in a *conventional state realization* of a linear code $\mathcal{C}$, the symbol index set $\mathcal{I}_\mathcal{A}$ is a conventional discrete time axis, namely the set of integers $\mathbb{Z}$, or a subinterval of $\mathbb{Z}$. The state index set $\mathcal{I}_\mathcal{S}$ may be thought of as the set of times that occur *between* consecutive pairs of times in $\mathcal{I}_\mathcal{A}$, and the state time preceding symbol time $k \in \mathcal{I}_\mathcal{A}$ is conventionally also denoted by $k \in \mathcal{I}_\mathcal{S}$. The constraint codes $\{\mathcal{C}_k : k \in I_\mathcal{A}\}$ are linear codes indexed by the symbol index set $\mathcal{I}_\mathcal{A}$, and specify the set of all valid (s_k, a_k, s_{k+1}) transitions; *i.e.,* for each $k \in \mathcal{I}_\mathcal{A}$, $\mathcal{C}_k$ is a subspace of the vector space $S_k \times A_k \times S_{k+1}$. The full behavior $\mathfrak{B}$ of the realization is the set of all symbol/state trajectories

$(\mathbf{a}, \mathbf{s})$ such that (s_k, a_k, s_{k+1}) is a valid transition in $\mathcal{C}_k$ for all $k \in \mathcal{I}_\mathcal{A}$. The code $\mathcal{C}$ generated by the realization is the set of all symbol trajectories $\mathbf{a}$ that appear in some $(\mathbf{a}, \mathbf{s}) \in \mathfrak{B}$.

A *normal behavioral realization* is defined as a behavioral realization in which every symbol alphabet is involved in precisely one constraint code, and every state space is involved in precisely two constraint codes. Thus a conventional state realization is normal. As shown in [5], any behavioral realization may be straightforwardly converted to a normal realization by the "normalization" procedure that will be described in the next subsection, without essentially increasing the complexity of the realization.

A normal behavioral realization has a natural graphical model, called a *normal graph*, in which each constraint code $\mathcal{C}_i$ corresponds to a vertex, each state space S_j (which by definition is involved in two constraints) corresponds to an edge connecting the two corresponding constraint vertices, and each symbol alphabet A_k (which by definition is involved in one constraint) corresponds to a leaf or "half-edge" connected to the corresponding constraint vertex. For example, Figure 1 shows the normal graph corresponding to a conventional state realization, which is a simple chain graph. Here vertices are represented by square boxes, and the "half-edges" corresponding to symbol alphabets are represented by special "dongle" symbols.

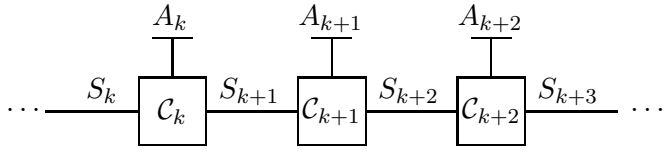

Figure 1: Normal graph of a conventional state realization.

2.2 Normal realizations and normal factor graphs

In this paper, we will mostly represent codes (and weight generating functions of codes) by normal factor graphs and their partition functions. A *partition function* will be defined by

- a set $\mathbf{A} = \{A_k, k \in \mathcal{I}_\mathcal{A}\}$ of *external variables* A_k with alphabets $\mathcal{A}_k$;
- a set $\mathbf{S} = \{S_j, j \in \mathcal{I}_\mathcal{S}\}$ of *internal variables* S_j with alphabets $\mathcal{S}_j$;
- a set $\mathbf{f} = \{f_i, i \in \mathcal{I}_\mathcal{F}\}$ of complex-valued *factors* f_i, each factor f_i involving subsets $\mathbf{A}_i \subseteq \mathbf{A}$ and $\mathbf{S}_i \subseteq \mathbf{S}$ of the sets of internal and external variables.

All sets are assumed to be finite, and all variable alphabets are assumed to be finite-dimensional vector spaces over some finite field $\mathbb{F}$. We call the Cartesian-product alphabet $\mathcal{A} = \prod_k \mathcal{A}_k$ the *external variable configuration space*, and $\mathcal{S} = \prod_j \mathcal{S}_j$ the *internal variable configuration space*.

The partition function[1] (or "external function" [1]) defined by these elements is the function $Z : \mathcal{A} \to \mathbb{C}$ of the external variables that is given by the following sum of products:

$$Z(\mathbf{a}) = \sum_{\mathbf{s} \in \mathcal{S}} \prod_{i \in \mathcal{I}_\mathcal{F}} f_i(\mathbf{a}_i, \mathbf{s}_i), \quad \mathbf{a} \in \mathcal{A};$$

i.e., the sum over all internal variable configurations of the product of all factors.

[1] In physics, a partition function is usually defined as a sum over internal configurations, and there are no external variables, so this usage of "partition function" extends the usual terminology of physics.

A partition function $Z(\mathbf{a})$ may in general be given by many different sum-of-products forms, which we will call *realizations*. We will say that two realizations of the same partition function are *equivalent*.

A realization will be called *normal* if each external variable is involved in precisely one factor, and each internal variable is involved in precisely two factors. As noted in [5], any realization may be converted to an equivalent normal realization by the following simple **normalization procedure**:

- For every external variable A_i, if A_i is involved in p factors, then define p *replica variables* $A_{i\ell}, 1 \le \ell \le p$, replace A_i by $A_{i\ell}$ in the ℓth factor in which A_i is involved, and introduce one new factor, namely an *equality indicator function* $\Phi_=(a_i, \{a_{i\ell}, 1 \le \ell \le p\})$; *i.e.,* the $(0,1)$-valued function that equals 1 when $a_i = a_{i1} = \cdots = a_{ip}$, and that equals 0 otherwise. Thus each replica variable $A_{i\ell}$ becomes an internal variable that is involved in precisely two factors, while A_i remains an external variable that is involved in only one factor, namely the equality indicator function. (If $p = 1$, then this conversion need not be performed.)

- For every internal variable S_j, if S_j is involved in $q \ge 2$ factors, then define q replica variables $S_{j\ell}, 1 \le \ell \le q$, replace S_j by $S_{j\ell}$ in the ℓth factor in which S_j is involved, and introduce one new factor, namely an equality indicator function $\Phi_=(\{s_{j\ell}, 1 \le \ell \le q\})$. Thus each replica variable $S_{j\ell}$ becomes an internal variable that is involved in precisely two factors. (If $q = 2$, then this conversion need not be performed. If $q = 1$, then multiply the partition function by a dummy factor $1(s_k)$ which is equal to 1 regardless of the value of S_k.)

Evidently this normalization procedure does not change the partition function $Z(\mathbf{a})$. Also, as can be seen from [5], the normal factor graph that represents the normal realization is essentially unchanged from the bipartite factor graph that represents the original realization.

A normal realization is represented by a *normal factor graph* (NFG).[2] As in a normal graph, ordinary edges represent internal variables and half-edges represent external variables, but now vertices represent factors rather than constraints. The partition function of the NFG is the partition function $Z(\mathbf{a})$ of the associated realization.

In this paper we will mostly adopt the convention of ignoring multiplicative scale factors $\alpha > 0$ in partition functions. We will say that the partition function is equal to $Z(\mathbf{a})$, *up to scale*; that is, the relative weights $Z(\mathbf{a})$ for the various configurations $\mathbf{a} \in \mathcal{A}$ are correct, but the absolute value may not be. In many applications the absolute scale factor is not important; but if it is, then it can be reconstructed from the constituent factors f_i.

A normal graph representing a code $\mathcal{C}$ may be converted to a normal factor graph representing the *indicator function* $\Phi_\mathcal{C}$ of $\mathcal{C}$ as follows. Let each local constraint code $\mathcal{C}_i$ be replaced by the $(0,1)$-valued indicator function $\Phi_{\mathcal{C}_i}$ of $\mathcal{C}_i$. Then, for a given external variable assignment $\mathbf{a} \in \mathcal{A}$, the partition function of the graph— *i.e.,* the sum over all $\mathbf{s} \in \mathcal{S}$ of the product of all local constraint code indicator functions— is the number of internal variable assignments $\mathbf{s} \in \mathcal{S}$ such that $(\mathbf{a}, \mathbf{s})$ satisfies all local constraints. For a linear code $\mathcal{C}$, it is often true that a unique $\mathbf{s} \in \mathcal{S}$ is determined by each $\mathbf{a} \in \mathcal{C}$; *e.g.,* when the graph is cycle-free and the realization is minimal. But in any case, by linearity, the same number of state sequences correspond to every $\mathbf{a} \in \mathcal{C}$, namely

[2] Loeliger *et al.* [12, 13] define "Forney-style factor graphs" just as we have defined normal factor graphs. However, in the usual factor graph framework, such a graph represents simply the product of the factors, rather than a sum of products. Loeliger *et al.* do also consider sums of products within "boxes," namely graph fragments enclosed by dashed lines.

$|\{\mathbf{s} \in \mathcal{S} : (\mathbf{0}, \mathbf{s}) \in \mathfrak{B}\}|$. Therefore, up to scale, the partition function of the NFG is equal to the $(0,1)$-valued indicator function $\Phi_{\mathcal{C}}$; *i.e.,*

$$\Phi_{\mathcal{C}}(\mathbf{a}) \propto \sum_{\mathbf{s} \in \mathcal{S}} \prod_{i \in \mathcal{I}_{\mathcal{C}}} \Phi_{\mathcal{C}_i}(\mathbf{a}_i, \mathbf{s}_i), \quad \mathbf{a} \in \mathcal{A}.$$

For example, Figure 2 shows the normal factor graph corresponding to the normal graph of Figure 1.

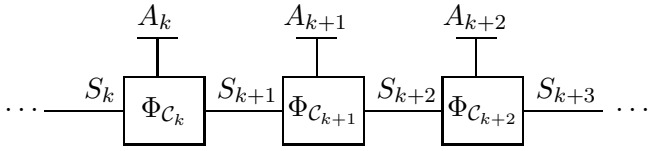

Figure 2: Normal factor graph of a conventional state realization.

2.3 Weight generating functions

Weight generating functions are standard tools of combinatorics. We will later consider various kinds of weight generating functions, but for linear codes over finite fields, the simplest and commonest kind of weight generating functions are Hamming weight generating functions.

The *Hamming weight generating function* (HWGF) of a linear code $\mathcal{C}$ defined over a finite index set $\mathcal{I}_{\mathcal{A}}$ with symbol sequence space $\mathcal{A} = \prod_{k \in \mathcal{I}_{\mathcal{A}}} A_k$ is the polynomial

$$g_{\mathcal{C}}^H(x) = \sum_{\mathbf{a} \in \mathcal{C}} \prod_{k \in \mathcal{I}_{\mathcal{A}}} x^{w_H(a_k)},$$

where $w_H(a_k)$ is the Hamming weight of the symbol $a_k \in A_k$. Thus the coefficient of x^w in $g_{\mathcal{C}}^H(x)$ is the number of codewords $\mathbf{a} \in \mathcal{C}$ that have Hamming weight w.

This sum-of-products expression for $g_{\mathcal{C}}^H(x)$ suggests that a HWGF might be represented as the partition function of a normal factor graph. (This idea was briefly mentioned in [11, Example 13].) Indeed, since

$$g_{\mathcal{C}}^H(x) = \sum_{\mathbf{a} \in \mathcal{A}} \Phi_{\mathcal{C}}(\mathbf{a}) x^{w_H(\mathbf{a})},$$

it follows that $g_{\mathcal{C}}^H(x)$ is the partition function of the simple normal factor graph of Figure 3(a), in which the two functions $\Phi_{\mathcal{C}}(\mathbf{a})$ and $x^{w_H(\mathbf{a})}$ are connected by the internal variable $\mathcal{A}$.

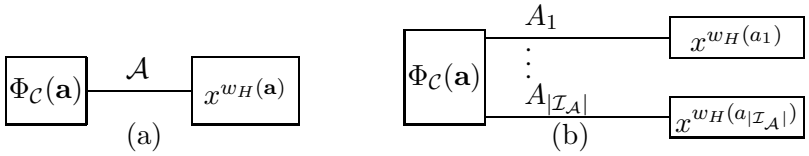

Figure 3: Normal factor graphs of the HWGF $g_{\mathcal{C}}^H(x)$ of a code $\mathcal{C} \subseteq \mathcal{A}$.

Or, since $w_H(\mathbf{a}) = \sum_{k \in \mathcal{I}_{\mathcal{A}}} w_H(a_k)$ is an additive function, $g_{\mathcal{C}}^H(x)$ may alternatively be represented as the partition function of the normal factor graph of Figure 3(b), in which the function $\Phi_{\mathcal{C}}(\mathbf{a})$ is connected by an internal variable A_k to the function $x^{w_H(a_k)}$ for each $k \in \mathcal{I}_{\mathcal{A}}$.

Now, more generally, if the indicator function $\Phi_{\mathcal{C}}$ of a linear code $\mathcal{C}$ is the partition function of any normal factor graph, then a normal factor graph for $g_{\mathcal{C}}^H(x)$ may be obtained by replacing the symbol half-edge associated with A_k in that graph by an ordinary edge connected to the function $x^{w_H(a_k)}$ of A_k for each $k \in \mathcal{I}_{\mathcal{A}}$. (Or, if $w_H(a_k) = \sum_\ell w_H(a_{k\ell})$ for some components $a_{k\ell}$ of a_k, then A_k and $x^{w_H(a_k)}$ may be broken down into their components $A_{k\ell}$ and $x^{w_H(a_{k\ell})}$ as in Figure 3(b).)

For example, for a normal factor graph of a conventional state realization as in Figure 2, the HWGF $g_{\mathcal{C}}^H(x)$ is the partition function of the normal factor graph shown in Figure 4.

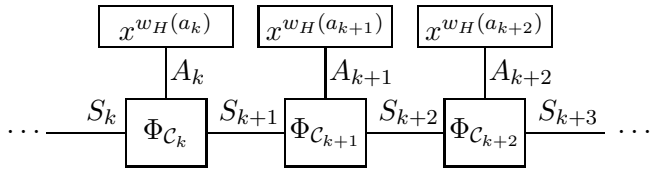

Figure 4: Normal factor graph of the HWGF $g_{\mathcal{C}}^H(x)$ of the code $\mathcal{C}$ of Figure 2.

Such a representation of $g_{\mathcal{C}}^H(x)$ may be further simplified by summing over the variables A_k for each $k \in \mathcal{I}_{\mathcal{A}}$, which have now become internal variables. For a code represented by a conventional state realization as in Figure 4, this results in a normal factor graph as in Figure 5. Here $\Lambda_{\mathcal{C}_k}^H(s_k, s_{k+1})$ is what is called [9, 10] the *Hamming weight adjacency matrix*[3] (HWAM) of $\mathcal{C}_k$:

$$\Lambda_{\mathcal{C}_k}^H(s_k, s_{k+1})(x) = \sum_{a_k \in A_k} \Phi_{\mathcal{C}_k}(s_k, a_k, s_{k+1}) x^{w_H(a_k)}$$

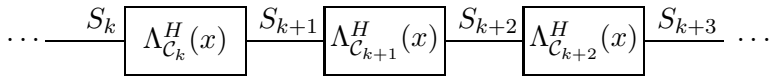

Figure 5: Normal factor graph resulting from summing over each A_k in Figure 4.

In equations, the equivalence of the partition functions of Figures 4 and 5 follows from

$$\begin{aligned} g_{\mathcal{C}}^H(x) &\propto \sum_{\mathbf{s}\in\mathcal{S}} \sum_{\mathbf{a}\in\mathcal{A}} \prod_{k\in\mathcal{I}_{\mathcal{A}}} \Phi_{\mathcal{C}_k}(s_k, a_k, s_{k+1}) x^{w_H(a_k)} \\ &= \sum_{\mathbf{s}\in\mathcal{S}} \prod_{k\in\mathcal{I}_{\mathcal{A}}} \sum_{a_k\in A_k} \Phi_{\mathcal{C}_k}(s_k, a_k, s_{k+1}) x^{w_H(a_k)} \\ &= \sum_{\mathbf{s}\in\mathcal{S}} \prod_{k\in\mathcal{I}_{\mathcal{A}}} \Lambda_{\mathcal{C}_k}^H(s_k, s_{k+1})(x). \end{aligned}$$

The last expression may be recognized as simply the product $\cdots \Lambda_{\mathcal{C}_k}(x)\Lambda_{\mathcal{C}_{k+1}}(x)\Lambda_{\mathcal{C}_{k+2}}(x)\cdots$ of the HWAMs, using the usual rules of matrix arithmetic.

Example 1 (binary linear block code). Consider the $(8, 4)$ binary linear first-order Reed-Muller code $\mathcal{C}$, which has the conventional four-section state realization (trellis) shown in Figure 6 [5].

[3]Perhaps this should have been called a "Hamming weight generating function adjacency matrix," but we prefer to follow the established terminology.

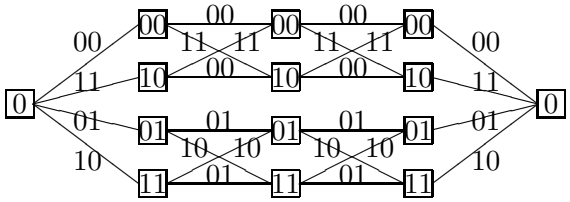

Figure 6: Four-section state realization of $(8, 4)$ binary first-order Reed-Muller code.

The Hamming weight generating function $g_{\mathcal{C}}^H(x)$ is given by the product of the HWAMs of each of the four sections, as follows:

$$\begin{bmatrix} 1 & x^2 & x & x \end{bmatrix} \begin{bmatrix} 1 & x^2 & 0 & 0 \\ x^2 & 1 & 0 & 0 \\ 0 & 0 & x & x \\ 0 & 0 & x & x \end{bmatrix} \begin{bmatrix} 1 & x^2 & 0 & 0 \\ x^2 & 1 & 0 & 0 \\ 0 & 0 & x & x \\ 0 & 0 & x & x \end{bmatrix} \begin{bmatrix} 1 \\ x^2 \\ x \\ x \end{bmatrix} = 1 + 14x^2 + x^8. \qquad \square$$

Let us now consider the Hamming weight generating function of a fragment of a normal factor graph realizing a code $\mathcal{C}$, consisting of a subset of the constraint codes $\mathcal{C}_i$. An external (symbol) variable A_k may or may not be involved in one of these constraint codes; if it is, we attach an enumerator function $x^{w_H(a_k)}$ as above. An internal (state) variable S_j may appear in two, one, or none of these constraint codes: if it appears twice, then we consider it to be an internal variable of the fragment and sum over it;[4] if it appears once, then we consider it to be an external variable of the fragment. In other words, the HWGF of the fragment is the partition function

$$g_f^H(x)(\mathbf{s}_{\text{ext}}) \propto \sum_{\mathbf{s}_{\text{int}}} \prod_{i \in \mathcal{I}_{\mathcal{A},f}} \sum_{\mathbf{a}_i} \Phi_{\mathcal{C}_i}(\mathbf{a}_i, \mathbf{s}_i) x^{w_H(\mathbf{a}_i)},$$

where $\mathbf{s}_{\text{ext}}$ and $\mathbf{s}_{\text{int}}$ are the external and internal state variables with respect to the fragment, and $\mathcal{I}_{\mathcal{A},f} \subseteq \mathcal{I}_{\mathcal{A}}$ is the subset of indices of constraint codes $\mathcal{C}_i$ that appear in the fragment.

Example 2 (N sections of a linear convolutional code). Consider a linear time-invariant convolutional code $\mathcal{C}$, in which all symbol alphabets A_k, state spaces S_k, and constraint codes $\mathcal{C}_k$ do not actually depend on the time index $k \in \mathbb{Z}$. Then the HWAM $\Lambda_{\mathcal{C}_k}^H(x)$ does not depend upon k, so we write it simply as $\Lambda(x)$. Consider a fragment of a graph for $\mathcal{C}$ consisting of N consecutive trellis sections, over a time interval $[k, k+N)$. Then, relative to this fragment, the external variables are S_k and S_{k+N}, and the HWAM of the fragment is simply the Nth power $\Lambda^N(x)$ of the HWAM $\Lambda(x)$ of a single section. $\square$

3 Duality

The central duality result of [5] is the *normal graph duality theorem*: given a normal realization of a code $\mathcal{C}$, the dual normal realization generates the dual (orthogonal) code $\mathcal{C}^\perp$. We will now give a concise proof of this result, as a corollary of a general duality theorem for normal factor graphs.

[4] Sometimes physicists use the Einstein summation convention: in a tensor product, variables that occur twice are implicitly to be summed over. This convention would evidently be useful for partition functions of normal factor graphs.

3.1 Fourier transforms

We first recall the theory of Fourier transforms over finite fields, generally following [4].

Let the prime p be the characteristic of the finite field $\mathbb{F}$; *i.e.,* p is the least positive integer such that $p\alpha = 0$ for all $\alpha \in \mathbb{F}$.

If A is a vector space over $\mathbb{F}$, then its dual space $\hat{A}$ may be defined as the set of all homomorphisms $\hat{a} : A \to \mathbb{Z}_p$. It follows from Pontryagin duality theory [4] that $\hat{A}$ is a vector space over $\mathbb{F}$ of the same dimension as A, and that the dual space to $\hat{A}$ is A, with $a(\hat{a})$ defined as $\hat{a}(a)$. We may then define the *inner product* $\langle \hat{a}, a \rangle$ as $a(\hat{a}) = \hat{a}(a)$ for all $a \in A, \hat{a} \in \hat{A}$. The inner product so defined has the usual properties; *e.g.,* $\langle \hat{a}, 0 \rangle = \langle 0, a \rangle = 0$, $\langle \hat{a}, a + a' \rangle = \langle \hat{a}, a \rangle + \langle \hat{a}, a' \rangle$, and so forth.

For example, if A is the field $\mathbb{F}$, then the additive group of $\mathbb{F}$ is isomorphic to $(\mathbb{Z}_p)^m$ for some integer m. The dual space $\hat{A}$ may also be taken as $\mathbb{F} \simeq (\mathbb{Z}_p)^m$, with the inner product defined componentwise as the dot product $\hat{a} \cdot a = \sum_{i=1}^{m} \hat{a}_i a_i$, with all operations in the prime field $\mathbb{Z}_p$.

Similarly, if A is the set $\mathbb{F}^n$ of all n-tuples over $\mathbb{F}$, then its dual space $\hat{A}$ may also be taken as $\mathbb{F}^n$, and the inner product may again be defined componentwise as

$$\langle \hat{a}, a \rangle = \sum_{j=1}^{n} \hat{a}_j \cdot a_j = \sum_{j=1}^{n} \sum_{i=1}^{m} \hat{a}_{ji} a_{ji}$$

Given a complex-valued function $f : A \to \mathbb{C}$ defined on A, its *Fourier transform* is defined as the complex-valued function $F : \hat{A} \to \mathbb{C}$ that maps $\hat{a} \in \hat{A}$ to

$$F(\hat{a}) = \sum_{a \in A} f(a) \omega^{\langle \hat{a}, a \rangle}, \quad \hat{a} \in \hat{A},$$

where $\omega = e^{2\pi i/p}$ is a primitive complex pth root of unity.

If we view $\mathbf{f} = \{f(a) : a \in A\}$ as a column vector indexed by A, and similarly $\mathbf{F} = \{F(\hat{a}) : \hat{a} \in \hat{A}\}$ as a column vector indexed by $\hat{A}$, then the transform can be expressed in matrix form as

$$\mathbf{F} = \mathcal{F}_A \mathbf{f},$$

where the *Fourier transform matrix* $\mathcal{F}_A$ is defined as $\{\omega^{\langle \hat{a}, a \rangle} : \hat{a} \in \hat{A}, a \in A\}$. Note that $\mathcal{F}_A$ is symmetric; *i.e.,* $\mathcal{F}_A^T = \mathcal{F}_A$, where $\mathcal{F}_A^T$ denotes the transpose of $\mathcal{F}_A$.

In a normal factor graph, a Fourier transform may be simply represented as in Figure 7. The transform $F(\hat{a})$ is obtained by summing over A, which in this case amounts to a matrix multiplication. Note that as a factor in a factor graph, we do not have to distinguish between $\mathcal{F}_A$ and its transpose; $\mathcal{F}_A$ is simply a function of the two variables corresponding to the two incident edges, and as a matrix can act on either variable.

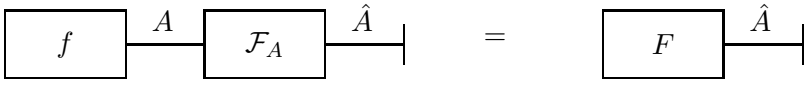

Figure 7: Normal factor graph of a Fourier transform.

From the *orthogonality relation*

$$\sum_{\hat{a} \in \hat{A}} \omega^{\langle \hat{a}, a \rangle} = \begin{cases} |A|, & a = 0; \\ 0, & a \neq 0, \end{cases}$$

we obtain the matrix equation

$$\mathcal{F}_A^* \mathcal{F}_A = |A| I_A,$$

where $\mathcal{F}_A^* = \{\omega^{-\langle \hat{a}, a \rangle} : a \in A, \hat{a} \in \hat{A}\}$ is the conjugate transpose of $\mathcal{F}_A$, and I_A is the identity matrix over A. In other words, the inverse of $\mathcal{F}_A$ is $\mathcal{F}_A^{-1} = |A|^{-1} \mathcal{F}_A^*$. Thus we obtain the *inverse Fourier transform*

$$\mathbf{f} = \mathcal{F}_A^{-1} \mathbf{F} = \frac{\mathcal{F}_A^* \mathbf{F}}{|A|}.$$

We say that $\mathbf{f}$ and $\mathbf{F}$ are a *Fourier transform pair*, written $\mathbf{f} \leftrightarrow \mathbf{F}$.

More generally, we have the following orthogonality relation (which even more generally applies to orthogonal subgroups of finite abelian groups):

Orthogonality relation for subspaces. Let B be any subspace of a finite subspace A, and let $B^\perp$ be the orthogonal subspace of its dual space $\hat{A}$; *i.e.,* $B^\perp$ is the subset of elements $\hat{a} \in \hat{A}$ such that $\langle \hat{a}, a \rangle = 0$ for all $a \in B$. Then

$$\sum_{a \in B} \omega^{\langle \hat{a}, a \rangle} = \begin{cases} |B|, & \hat{a} \in B^\perp; \\ 0, & \hat{a} \notin B^\perp. \end{cases}$$

Proof: Obvious for $\hat{a} \in B^\perp$. For $\hat{a} \notin B^\perp$, let $z(\hat{a}) = \sum_{a \in B} \omega^{\langle \hat{a}, a \rangle}$, and let a' be some element of B such that $\langle \hat{a}, a' \rangle \neq 0$. Since $a' + B = B$, we have $z(\hat{a}) = \sum_{a \in B} \omega^{\langle \hat{a}, a' + a \rangle} = \omega^{\langle \hat{a}, a' \rangle} z(\hat{a})$; but $\langle \hat{a}, a' \rangle \neq 0$ implies $\omega^{\langle \hat{a}, a' \rangle} \neq 1$, so $z(\hat{a}) = \omega^{\langle \hat{a}, a' \rangle} z(\hat{a})$ implies $z(\hat{a}) = 0$. □

Let $\Phi_B : A \to \{0, 1\}$ be the indicator function of the subspace B; *i.e.,* $\Phi_B(a) = 1$ if $a \in B$, else $\Phi_B(a) = 0$. From this orthogonality relation, it follows that the Fourier transform of Φ_B is $|B| \Phi_{B^\perp}$. Conversely, by a similar argument, the inverse Fourier transform of $\Phi_{B^\perp}$ is $|B^\perp| \Phi_B / |A|$. Thus if B and $B^\perp$ are orthogonal subspaces of a finite vector space A and its dual space $\hat{A}$, respectively, then

- the indicator functions Φ_B and $\Phi_{B^\perp}$ are a Fourier transform pair, up to scale;
- $|B||B^\perp| = |A| = |\hat{A}|$.

As [15] notes, the top result is a version of the "picket-fence miracle" [4] (see also [13, Theorem 9]).

We observe that the Poisson summation formula, namely

$$\sum_{\mathbf{a} \in B} f(\mathbf{a}) = |B^\perp| \sum_{\hat{\mathbf{a}} \in B^\perp} F(\hat{\mathbf{a}}),$$

follows directly from the Fourier transform relationship for orthogonal code indicator functions, where the functions $f \leftrightarrow F$ are any Fourier transform pair; see Figure 8. (Note that since both Φ_B and $\Phi_{B^\perp}$ are real-valued, the Fourier transform relation holds with the kernel $\mathcal{F}_A^*$ as well as with $\mathcal{F}_A$.) We conclude that the Fourier transform pair relationship $\Phi_B \leftrightarrow \Phi_{B^\perp}$ is more fundamental than the Poisson summation formula, which is the usual starting point in the development of MacWilliams identities.

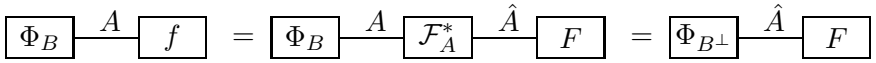

Figure 8: Proof of the Poisson summation formula.

Finally, we extend these definitions to a set of indeterminates $\mathbf{x} = \{x(a) : a \in A\}$ indexed by A, rather than a complex-valued function. The transform of this set is then a dual set of indeterminates $\mathbf{X} = \{X(\hat{a}) : \hat{a} \in \hat{A}\}$ indexed by $\hat{A}$, defined by

$$\mathbf{X} = \mathcal{F}_A \mathbf{x}.$$

Again, we have the inverse transform relationship

$$\mathbf{x} = \mathcal{F}_A^{-1} \mathbf{X} = \frac{\mathcal{F}_A^* \mathbf{X}}{|A|},$$

and we say that $\mathbf{x}$ and $\mathbf{X}$ are a transform pair, written $\mathbf{x} \leftrightarrow \mathbf{X}$. Figure 9 shows the corresponding normal factor graph, similar to Figure 7.

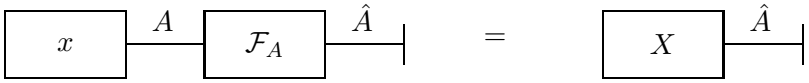

Figure 9: Normal factor graph of a set $\mathbf{x}$ of indeterminates indexed by A, and the dual set $\mathbf{X}$.

For example, if $A = \mathbb{Z}_2$, then $X(0) = x(0) + x(1)$ and $X(1) = x(0) - x(1)$; similarly, $x(0) = \frac{1}{2}(X(0) + X(1))$, and $x(1) = \frac{1}{2}(X(0) - X(1))$. For another example, if $A = (\mathbb{Z}_2)^2$, then

$$\begin{aligned} X(00) &= x(00) + x(10) + x(01) + x(11) \\ X(10) &= x(00) - x(10) + x(01) - x(11) \\ X(01) &= x(00) + x(10) - x(01) - x(11) \\ X(11) &= x(00) - x(10) - x(01) + x(11) \end{aligned}$$

and *vice versa* (up to a scale factor of $\frac{1}{4}$).

3.2 Dual linear codes

Again, a linear code $\mathcal{C}$ over a finite field $\mathbb{F}$ is a subspace $\mathcal{C} \subseteq \mathcal{A}$ of a symbol sequence space $\mathcal{A} = \prod_{k \in \mathcal{I}_\mathcal{A}} A_k$, where each symbol alphabet A_k is a finite-dimensional vector space over $\mathbb{F}$.

As we have seen, each vector space A_k has a dual space $\hat{A}_k$ of the same dimension such that for all $a_k \in A_k, \hat{a}_k \in \hat{A}_k$ there is a well-defined inner product $\langle \hat{a}_k, a_k \rangle$. The dual space to the symbol sequence space $\mathcal{A} = \prod_{k \in \mathcal{I}_\mathcal{A}} A_k$ is then the dual symbol sequence space $\hat{\mathcal{A}} = \prod_{k \in \mathcal{I}_\mathcal{A}} \hat{A}_k$, with the inner product defined componentwise by $\langle \hat{\mathbf{a}}, \mathbf{a} \rangle = \sum_{k \in \mathcal{I}_\mathcal{A}} \langle \hat{a}_k, a_k \rangle$ (where we continue to assume that $\mathcal{I}_\mathcal{A}$ is finite). The dual (orthogonal) code $\mathcal{C}^\perp \subseteq \hat{\mathcal{A}}$ to $\mathcal{C} \subseteq \mathcal{A}$ is then the set of all dual symbol sequences $\hat{\mathbf{a}} \in \hat{\mathcal{A}}$ such that $\langle \hat{\mathbf{a}}, \mathbf{a} \rangle = 0$ for all $\mathbf{a} \in \mathcal{C}$.

From our general orthogonality theorem, we have that the indicator functions $\Phi_\mathcal{C}$ and $\Phi_{\mathcal{C}^\perp}$ are a Fourier transform pair, up to scale, and that $|\mathcal{C}||\mathcal{C}^\perp| = |\mathcal{A}|$.

Since the inner product is defined componentwise, *i.e.,* $\langle \hat{\mathbf{a}}, \mathbf{a} \rangle = \sum_{k \in \mathcal{I}_\mathcal{A}} \langle \hat{a}_k, a_k \rangle$, it follows that the Fourier transform may be taken separately with respect to each variable A_k involved in $\mathcal{C}$. For example, consider the indicator function $\Phi_{\mathcal{C}_k}$ of a constraint code $\mathcal{C}_k$ in a conventional state realization, as in Figure 2. The indicator function for the orthogonal code $\mathcal{C}_k^\perp \subseteq \hat{S}_k \times \hat{A}_k \times \hat{S}_{k+1}$ may be obtained (up to scale) by transforming with respect to each of the three incident variables separately, as shown in the normal factor graph of Figure 10.

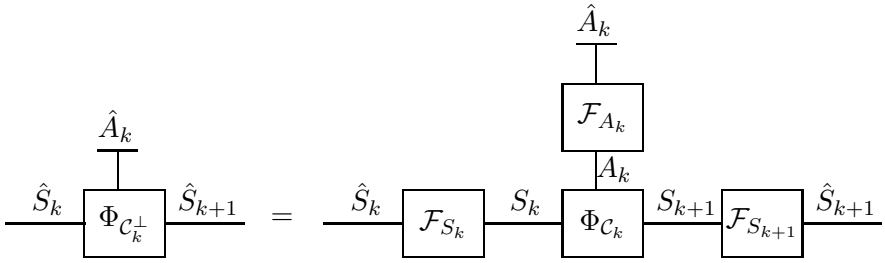

Figure 10: Transform of constraint code indicator function $\Phi_{\mathcal{C}_k}$ in a conventional state realization.

3.3 Fourier transform identities

In this subsection we develop a few quick and easy facts about the algebra of Fourier transforms.

We have already noted that $\mathcal{F}_A\mathcal{F}_A^* = I_A$, up to a scale factor of $|A|$. We will write such a relationship as $\mathcal{F}_A\mathcal{F}_A^* \propto I_A$. (Alternatively, we could normalize each Fourier transform matrix by an appropriate scale factor, namely $1/\sqrt{|A|}$ for $\mathcal{F}_A$.)

We next observe that

$$\sum_{\hat{a}\in\hat{A}} \omega^{\langle\hat{a},a\rangle}\omega^{\langle\hat{a},a'\rangle} = \begin{cases} |A|, & a+a'=0; \\ 0, & a+a'\neq 0, \end{cases}$$

by the basic orthogonality relation. In other words, $\mathcal{F}_A^T\mathcal{F}_A = |A|\Phi_{\sim A}$, where $\Phi_{\sim A}$ is the *sign inverter indicator function* on A; *i.e.,* the indicator function of the sign inversion relation, $a' = -a$.

More simply, as a factor in a normal factor graph, the concatenation of $\mathcal{F}_A$ with itself is equivalent up to scale to $\Phi_{\sim A}$, provided that the alphabet at each end of the chain is A; see the top line of Figure 11. In this sense, we may write $(\mathcal{F}_A)^2 \propto \Phi_{\sim A}$, where the product indicates concatenation. (Note that if the alphabet at each end of the chain were $\hat{A}$, then $(\mathcal{F}_A)^2$ would be equivalent up to scale to $\Phi_{\sim\hat{A}}$, the sign inverter indicator function on $\hat{A}$.)

In the same sense, it is easy to see that $(\mathcal{F}_A)^3 \propto \mathcal{F}_A^*$, and $(\mathcal{F}_A)^4 \propto I_A$. In other words, $\mathcal{F}_A$ is a fourth root of unity under concatenation, up to scale, and indeed behaves very much like $i = \sqrt{-1}$. Similarly, $\mathcal{F}_A^* \propto (\mathcal{F}_A)^3$ is a fourth root of unity that is conjugate to $\mathcal{F}_A$, while $\Phi_{\sim A} \propto (\mathcal{F}_A)^2$ is a square root of unity and is real (equal to its conjugate).

Figure 11 illustrates these relationships. In this figure, edges are labelled simply by the alphabet of the associated variable rather than by the variable itself.

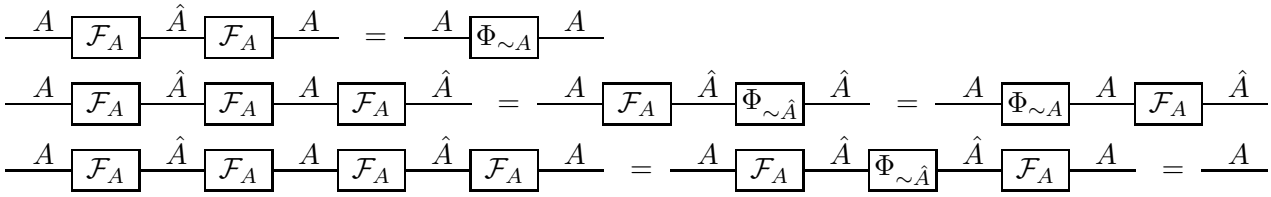

Figure 11: Fourier transform identities.

We remark that the relation $I_A \propto (\mathcal{F}_A)^4 \propto \mathcal{F}_A\Phi_{\sim\hat{A}}\mathcal{F}_A$ is an instance of the Fourier transform relationship for orthogonal code indicator functions, since for any vector space A and its dual $\hat{A}$, the orthogonal code to the sign inverter code $\mathcal{C}_{\sim\hat{A}} = \{(\hat{a}, -\hat{a}) : \hat{a} \in \hat{A}\}$, whose indicator function is $\Phi_{\sim\hat{A}}$, is the repetition code $\mathcal{C}_{=A} = \{(a, a) : a \in A\}$, whose indicator function is I_A.

Finally, note that in the special case where the characteristic of $\mathbb{F}$ is $p = 2$, we have that $\mathcal{F}_A^* = \mathcal{F}_A$ and $\Phi_{\sim A} = I_A$; *i.e.,* $\mathcal{F}_A$ becomes a square root of unity, and the sign inverter relation becomes the equality relation, $a' = a$.

3.4 Normal factor graph duality theorem for linear codes

We will now prove the normal graph duality theorem for linear codes by dualizing a normal factor graph. We continue to ignore scale factors.

Again, we start with a normal factor graph whose partition function is the indicator function $\Phi_{\mathcal{C}}$ of a linear code $\mathcal{C} \subseteq \mathcal{A}$, with half-edges representing external variables $\{A_k\}$, edges representing internal variables $\{S_j\}$, and vertices representing indicator functions $\{\Phi_{\mathcal{C}_i}\}$.

To obtain a normal factor graph whose partition function is the indicator function $\Phi_{\mathcal{C}^\perp}$ of the orthogonal linear code $\mathcal{C}^\perp \subseteq \hat{\mathcal{A}}$, we apply appropriate Fourier transforms $\mathcal{F}$ to each of the external variables A_k, as in Figure 9. For example, Figure 12 illustrates this transformation for two consecutive sections of a conventional state realization as in Figure 2.

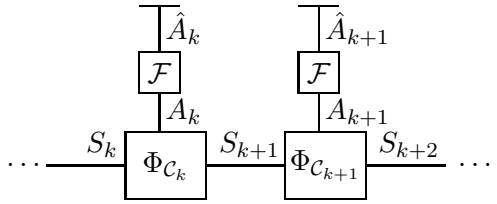

Figure 12: Normal factor graph representing the orthogonal code to that of Figure 2.

Next, using the concatenation relation $I \propto \mathcal{F}\Phi_\sim\mathcal{F}$ illustrated in Figure 11, we replace each edge by the concatenation of an appropriate transform matrix $\mathcal{F}$, sign inverter indicator function $\Phi_\sim$, and transform matrix $\mathcal{F}$. For example, Figure 13 illustrates this replacement for the normal factor graph of Figure 12. Here S_k and S'_k are two equal state variables, and dual state variables on opposite sides of sign inverters have been given opposite signs.

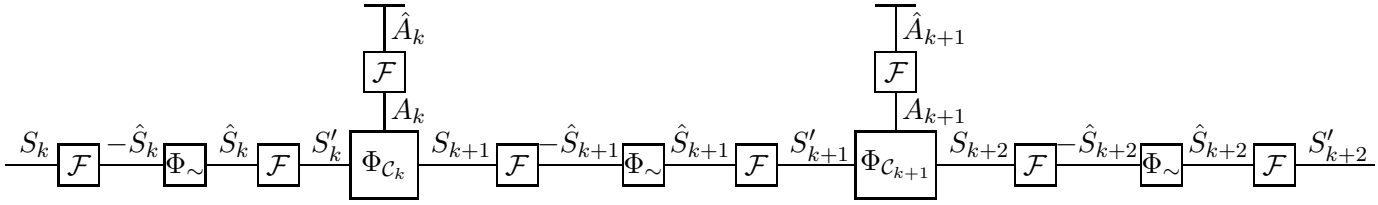

Figure 13: Second step in dualizing Figure 2.

We then observe that every constraint code indicator function is now surrounded by Fourier transforms, as in Figure 10, and therefore we may replace it by the indicator function of the dual constraint code, as shown in Figure 14. Note that the sign inverter indicator functions remain.

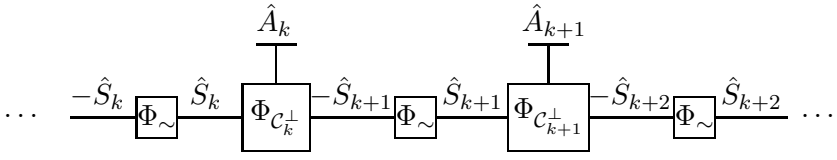

Figure 14: Final step in dualizing Figure 2.

In summary, we have proved:

Normal factor graph duality theorem for linear codes. Given a normal factor graph whose partition function is the indicator function $\Phi_{\mathcal{C}}$ of a linear code $\mathcal{C}$, comprising symbol alphabets $\{A_k\}$ associated with half-edges, state spaces $\{S_j\}$ associated with ordinary edges, and constraint code indicator functions $\{\Phi_{\mathcal{C}_i}\}$ associated with vertices, the dual normal factor graph is defined by replacing each symbol alphabet A_k by its dual symbol alphabet $\hat{A}_k$, each state space S_k by its dual state space $\hat{S}_k$, each indicator function $\Phi_{\mathcal{C}_i}$ by the dual indicator function $\Phi_{\mathcal{C}_i^\perp}$, and finally by placing a sign inverter indicator function $\Phi_\sim$ in the middle of every ordinary edge. Then the partition function of the dual normal factor graph is equal to $\Phi_{\mathcal{C}^\perp}$, up to scale. □

This theorem is equivalent to our original normal graph duality theorem [5]: given a normal graph representing a linear code $\mathcal{C}$, comprising symbol alphabets $\{A_k\}$ associated with half-edges, state spaces $\{S_j\}$ associated with ordinary edges, and constraint codes $\{\mathcal{C}_i\}$ associated with vertices, the dual normal graph is defined by replacing each symbol alphabet A_k by its dual symbol alphabet $\hat{A}_k$, each state space S_k by its dual state space $\hat{S}_k$, each constraint code $\mathcal{C}_i$ by its orthogonal code $\mathcal{C}_i^\perp$, and finally by placing a sign inverter in the middle of every ordinary edge. Then the dual normal graph represents the orthogonal code $\mathcal{C}^\perp$.

For example, Figure 15 shows the dual to the normal graph of a conventional state realization that was shown in Figure 1.

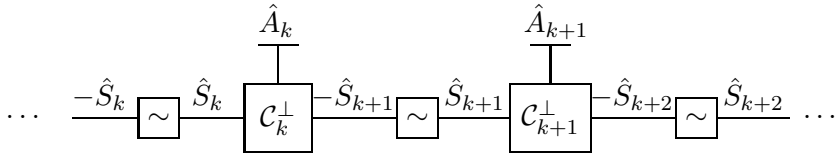

Figure 15: Dual normal graph for a conventional state realization.

The following two examples illustrate how to dualize binary and nonbinary convolutional codes by dualizing the local constraint codes $\mathcal{C}_k$. Example 3 involves a rate-1/2, four-state binary convolutional code that is often used in examples. Example 4 involves the rate-2/3 ternary convolutional code that was the principal example of Gluesing-Luerssen and Schneider [9, 10].

Example 3 (binary linear convolutional code). Consider the rate-1/2 binary linear time-invariant convolutional code $\mathcal{C}$ generated by the degree-2 generators $(1 + D^2, 1 + D + D^2)$, in standard D-transform notation. In other words, $\mathcal{C}$ is the set of all output sequences of the single-input, two-output linear time-invariant system over $\mathbb{F}_2$ whose impulse response is $(11, 01, 11, 00, \ldots)$. This system has a conventional state realization as in Figure 1 in which each symbol alphabet A_k may be taken as $(\mathbb{F}_2)^2$, each state space S_k may also be taken as $(\mathbb{F}_2)^2$, and each constraint code $\mathcal{C}_k$ is the $(6, 3)$ binary linear block code generated by the three transitions

$$\begin{array}{ccc} 00 & 11 & 10 \\ 10 & 01 & 01 \\ 01 & 11 & 00 \end{array}$$

which represent the three nontrivial (state, symbol, next state) transitions in the impulse response of the system. (Note that only the output symbols appear in $\mathcal{C}$; the input symbols that would appear in an input-state-output realization are here regarded as internal variables, and do not appear explicitly.) The eight codewords of $\mathcal{C}_k$ are the eight possible transitions of the system, which are shown as a "trellis section" in Figure 16(a), with the three generating transitions dashed.

Figure 16: Trellis sections of (a) rate-1/2 binary convolutional code; (b) orthogonal code.

The orthogonal code $\mathcal{C}_k^\perp$ may easily be seen to be the $(6, 3)$ binary linear block code generated by the three transitions

$$\begin{array}{ccc} 00 & 11 & 01 \\ 01 & 10 & 10 \\ 10 & 11 & 00 \end{array}$$

which represent the three nontrivial (state, symbol, next state) transitions in the impulse response of a system with impulse response $(11, 10, 11, 00, \ldots)$, or $(1+D+D^2, 1+D^2)$ in D-transform notation. This is indeed the generator of the orthogonal convolutional code $\mathcal{C}^\perp$ under the symbolwise definition of the inner product that we are using here. (For the more usual sequencewise definition of the inner product, we need to take the time reversal of $\mathcal{C}^\perp$,[5] which in this case happens to be the code generated by $(1 + D + D^2, 1 + D^2)$ again.) The eight codewords of $\mathcal{C}_k^\perp$ are the eight possible transitions of the orthogonal system, which are again shown as a "trellis section" in Figure 16(b), with the three generating transitions dashed. □

Example 4 (ternary linear convolutional code; *cf.* [9, 10]). Consider the rate-2/3 linear time-invariant convolutional code $\mathcal{C}$ over $\mathbb{F}_3$ with $g_1(D) = (1 + D^2, 2 + D, 0)$ and $g_2(D) = (1, 0, 2)$. In other words, $\mathcal{C}$ is the set of all output sequences of the two-input, three-output linear time-invariant system over $\mathbb{F}_3$ whose impulse responses are $(120, 010, 100, 000, \ldots)$ and $(102, 000, \ldots)$. This system has a conventional nine-state realization as in Figure 1 in which each symbol alphabet A_k may be taken as $(\mathbb{F}_3)^3$, each state space S_k may be taken as $(\mathbb{F}_3)^2$, and each constraint code $\mathcal{C}_k$ is the $(7, 4)$ ternary linear block code generated by the four generators

$$\begin{array}{ccc} 00 & 120 & 10 \\ 10 & 010 & 01 \\ 01 & 100 & 00 \\ 00 & 102 & 00 \end{array}$$

which represent the four nontrivial (s_k, a_k, s_{k+1}) transitions in the two impulse responses of the system. The orthogonal code $\mathcal{C}_k^\perp$ is the $(7, 3)$ ternary linear block code generated by the three generators

$$\begin{array}{ccc} 00 & 010 & 12 \\ 21 & 202 & 11 \\ 22 & 111 & 00 \end{array}$$

[5]The symbolwise inner product of two sequences $\mathbf{a}, \mathbf{b} \in \mathcal{A}$ is $\sum_k a_k b_k$, and that of $\mathbf{a}$ and a shift of $\mathbf{b}$ by j time units is $\sum_k a_k b_{k-j}$. The product of the corresponding D-transforms $a(D) = \sum_k a_k D^k$ and $b(D^{-1}) = \sum_k b_k D^{-k}$ is $\sum_j (\sum_k a_k b_{k-j}) D^j$, so $\mathbf{a}$ is orthogonal to all shifts of $\mathbf{b}$ if and only if $a(D)b(D^{-1}) = 0$, or equivalently if and only if the sequencewise inner product $a(D)\tilde{b}(D)$ is zero, where $\tilde{b}(D)$ is the D-transform of the time-reversed sequence $\tilde{\mathbf{b}} = \{b_{-k} : k \in \mathcal{I}_\mathcal{A}\}$. Thus $\mathcal{D}$ is the orthogonal code to a linear time-invariant code $\mathcal{C}$ under the symbolwise inner product if and only if the time-reversed code $\tilde{\mathcal{D}}$ is the orthogonal code to $\mathcal{C}$ under the sequencewise inner product.

which represent the three nontrivial $(\hat{s}_k, \hat{a}_k, -\hat{s}_{k+1})$ transitions in the impulse response of a conventional state realization of a single-input, three-output linear system over $\mathbb{F}_3$, with sign inverters, whose impulse response is $(010, 202, 111, 000, \ldots)$, or $(2D + D^2, 1 + D^2, 2D + D^2)$ in D-transform notation. (Note the unconventional basis of the dual state space, and the effects of the sign inversions.) This is indeed the generator of the orthogonal convolutional code $\mathcal{C}^\perp$ under our symbolwise definition of the inner product. (For the more usual sequencewise definition of the inner product, we need to take the time reversal of $\mathcal{C}^\perp$, which in this case is the code generated by $(1 + 2D, 1 + D^2, 1 + 2D)$.) □

3.5 General normal factor graph duality theorem

Finally, we note that although we have been focussing on normal factor graphs whose factors and partition functions are indicator functions of linear codes, the proof of the normal factor graph duality theorem in the previous subsection does not depend upon this restriction. Thus we have actually proved a much more general theorem:

General normal factor graph duality theorem. Given an arbitrary normal factor graph with partition function $Z(\mathbf{a})$, up to scale, comprising external variables $\{A_k\}$ associated with half-edges, internal variables $\{S_j\}$ associated with ordinary edges, and factors $\{f_i\}$ associated with vertices, the dual normal factor graph is defined by replacing each external variable A_k by its dual variable $\hat{A}_k$, each internal variable S_j by its dual variable $\hat{S}_j$, each factor f_i by its Fourier transform $\hat{f}_i$, and finally by placing a sign inverter indicator function $\Phi_\sim$ in the middle of every ordinary edge. Then the partition function of the dual normal factor graph is the Fourier transform $\hat{Z}(\hat{\mathbf{a}})$ of $Z(\mathbf{a})$, up to scale. (Note: [1] shows that the scale factor is $\prod_j |\mathcal{S}_j|$.) □

Mao and Kschischang [15] have proved a closely related duality theorem. In their development, a (multiplicative) factor graph represents a product of factors, rather than a sum of products. They show that the dual to a multiplicative factor graph is a *convolutional* factor graph, which represents the convolution of its factors. If the elements of a factor graph are dualized as above (but without the sign inverters), then a factor graph and its dual represent a Fourier transform pair, up to scale. Using this duality result and the fact that the indicator functions of a linear code and its dual are a Fourier transform pair, they derive a factor graph duality theorem for linear codes. They then specialize to normal factor graphs, and show how to recover our original normal graph duality theorem for linear codes. A similar derivation would yield the general normal factor graph duality theorem above. The advantage of our approach is that by focussing on normal factor graphs and their partition functions from the beginning, we do not need to introduce convolutional factor graphs (which may however prove to have their uses).

Al-Bashabsheh and Mao [1] have also proved this theorem by their methods, independently and at about the same time. Loeliger *et al.* [13, Appendix III] have proved an analogous theorem for continuous alphabets.

4 MacWilliams identities

Given these duality results, we can obtain various MacWilliams identities, using similar derivations.

4.1 MacWilliams identities for exact weight generating functions

We will first consider what MacWilliams and Sloane [14] call *exact* weight generating functions, where every symbol alphabet A_k is given its own set of indeterminates $\mathbf{x}_k = \{x_k(a_k) : a_k \in A_k\}$. The dual set of indeterminates $\mathbf{X}_k = \{X_k(\hat{a}_k) : \hat{a}_k \in \hat{A}_k\}$ is then given by the transform $\mathbf{X}_k = \mathcal{F}_{A_k}\mathbf{x}_k$.

The *exact weight generating function* (EWGF) of $\mathcal{C}$ is the multivariate polynomial

$$g^E_{\mathcal{C}}(\mathbf{x}) = \sum_{\mathbf{a}\in\mathcal{C}} \mathbf{x}(\mathbf{a}) = \sum_{\mathbf{a}\in\mathcal{A}} \Phi_{\mathcal{C}}(\mathbf{a})\mathbf{x}(\mathbf{a}),$$

where $\mathbf{x}(\mathbf{a}) = \prod_{k\in\mathcal{I}_{\mathcal{A}}} x_k(a_k)$.

Similarly, the EWGF of $\mathcal{C}^\perp$ is the multivariate polynomial

$$g^E_{\mathcal{C}^\perp}(\mathbf{X}) = \sum_{\hat{\mathbf{a}}\in\hat{\mathcal{A}}} \Phi_{\mathcal{C}^\perp}(\hat{\mathbf{a}})\mathbf{X}(\hat{\mathbf{a}}).$$

By the Poisson summation formula, these dual EWGFs are equal, up to scale; see Figure 17. (Again we note that since both $\Phi_{\mathcal{C}}$ and $\Phi_{\mathcal{C}^\perp}$ are real-valued, the Fourier transform relation holds with the kernel $\mathcal{F}^*_{\mathcal{A}}$ as well as with $\mathcal{F}_{\mathcal{A}}$.) Explicitly, we have the following MacWilliams identity showing how the EWGF of $\mathcal{C}^\perp$ may be computed from that of $\mathcal{C}$:

$$g^E_{\mathcal{C}^\perp}(\mathbf{X}) \propto g^E_{\mathcal{C}}(\mathcal{F}^*_{\mathcal{A}}\mathbf{X})$$

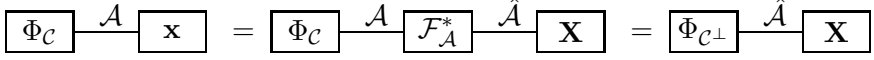

Figure 17: Duality of exact weight generating functions.

Now again suppose that we have a normal factor graph whose partition function is $\Phi_{\mathcal{C}}$ (up to scale), defined by a set $\mathbf{A} = \{A_k\}$ of symbol alphabets, a set $\mathbf{S} = \{S_j\}$ of state spaces, and a set $\{\Phi_{\mathcal{C}_i}\}$ of constraint code indicator functions, each constraint code $\mathcal{C}_i$ constraining subsets $\mathbf{A}_i \subseteq \mathbf{A}, \mathbf{S}_i \subseteq \mathbf{S}$ of the symbol and state alphabets, respectively. Then we may obtain a normal factor graph whose partition function is the global EWGF $g^E_{\mathcal{C}}(\mathbf{x})$ by connecting each half-edge representing A_k to a corresponding indeterminate function $x_k : a_k \mapsto x_k(a_k)$, as in Figure 3. The *local exact weight generating function* $g^E_{\mathcal{C}_i}(\mathbf{x}_i, \mathbf{s}_i)$ is then obtained by summing over $\mathbf{A}_i$ for each $\mathcal{C}_i$ to obtain

$$g^E_{\mathcal{C}_i}(\mathbf{x}_i, \mathbf{s}_i) = \sum_{\mathbf{a}_i\in\mathbf{A}_i} \Phi_{\mathcal{C}_i}(\mathbf{a}_i, \mathbf{s}_i)\mathbf{x}_i(\mathbf{a}_i).$$

The global EWGF is then the resulting global partition function, up to scale:

$$g^E_{\mathcal{C}}(\mathbf{x}) = \sum_{\mathbf{s}\in\mathcal{S}} \prod_{i\in\mathcal{I}_{\mathcal{C}}} g^E_{\mathcal{C}_i}(\mathbf{x}_i, \mathbf{s}_i).$$

In the case of a conventional state realization, the local EWGF is a matrix indexed by $S_k \times S_{k+1}$,

$$\Lambda_k^E(x_k) = \{g_{\mathcal{C}_k}^E(x_k, s_k, s_{k+1}) : s_k \in S_k, s_{k+1} \in S_{k+1}\},$$

which we will call the *exact weight adjacency matrix* (EWAM), and the global EWGF is the matrix product of these EWAMs, as was shown for HWGFs and HWAMs in Figures 4 and 5.

In general, if the graph of a code is a tree (*i.e.,* cycle-free), then since a global generating function of the code is a sum of products of local factors, it may be computed by a straightforward application of the generic sum-product algorithm [5], in which the "messages" are generating functions of subtrees. Even if the graph is not a tree, the global generating function is still the sum of the products of the local generating functions over all $\mathbf{s} \in \mathcal{S}$. Consequently one method of computing it is to cut just enough state edges so that the graph becomes a tree (a minimal spanning tree), compute the global generating function for this tree using the sum-product algorithm, and then sum over the remaining state variables.

Example 5 (tail-biting trellis). A tail-biting trellis consists of a chain of N trellis sections, with the further constraint that the final state s_N is equal to the initial state s_0. To compute a global generating function of a tail-biting trellis, we may thus first form the matrix product of the N constituent local WAMs to obtain a global WAM, indexed by (s_0, s_N); we then sum over all elements of this WAM for which $s_0 = s_N$; *i.e.,* we take the *trace* of the global WAM. □

To obtain a MacWilliams identity for local EWGFs, we proceed as follows. Each local EWGF $g_{\mathcal{C}_i}^E(\mathbf{x}_i, \mathbf{s}_i)$ is the partition function of the local graph fragment shown in Figure 18(a), comprising a local constraint code indicator function $\Phi_{\mathcal{C}_i}$ attached via an edge labelled by the local symbol alphabets $\mathbf{A}_i$ to an exact weight generator function $\mathbf{x}_i$. Here we partition the local state spaces $\mathbf{S}_i$ into two subsets $\mathbf{S}_i^+$ and $\mathbf{S}_i^-$, such that globally every state space S_j appears once in a plus subset and once in a minus subset (corresponding to the sign inversion in the dual normal factor graph). Thus the local EWGF will now be written as $g_{\mathcal{C}_i}^E(\mathbf{x}_i, \mathbf{s}_i^+, \mathbf{s}_i^-)$.

Proceeding again along the lines of the derivation of the Poisson summation formula (see Figure 8), we now replace the function $\mathbf{x}_i$ by the concatenation of $\mathbf{X}_i$ and an appropriate inverse Fourier transform $\mathcal{F}^*$; we adjoin appropriate inverse Fourier transforms $\mathcal{F}^*$ to each $S_j \in \mathbf{S}_i^+$; and we adjoin appropriate inverse Fourier transforms $\mathcal{F}^*$ and sign inverter indicator functions $\Phi_\sim$ to each $S_j \in \mathbf{S}_i^-$, as shown in Figure 18(b). Because the concatenation of $\mathcal{F}^*$, $\Phi_\sim$ and $\mathcal{F}^*$ is the identity (see Figure 11), and because globally every state space S_j appears once in a plus subset and once in a minus subset, this will leave the global EWGF unchanged.

Finally, noticing that each code constraint indicator function $\Phi_{\mathcal{C}_i}$ is now surrounded by (inverse) Fourier transforms, we may replace the whole ensemble by the orthogonal code indicator function $\Phi_{\mathcal{C}_i^\perp}$, as shown in Figure 18(c). Globally, it is evident that we now have a factor graph whose partition function is $g_{\mathcal{C}^\perp}^E(\mathbf{X})$, comprising the dual normal factor graph for $\Phi_{\mathcal{C}^\perp}$, with each half-edge representing $\hat{\mathbf{A}}_i$ connected to a corresponding indeterminate function $\mathbf{X}_i$.

Now we have the following MacWilliams identity, showing how the local EWGF of $\mathcal{C}_i^\perp$ may be computed from that of $\mathcal{C}_i$:

$$g_{\mathcal{C}_i^\perp}^E(\mathbf{X}_i, \hat{\mathbf{S}}_i^+, \hat{\mathbf{S}}_i^-) \propto g_{\mathcal{C}_i}^E(\mathcal{F}_{\mathbf{A}_i}^* \mathbf{X}_i, \mathcal{F}_{\mathbf{S}_i^+}^* \mathbf{S}_i^+, \mathcal{F}_{\mathbf{S}_i^-}(-\mathbf{S}_i^-)),$$

where $\mathcal{F}_{\mathbf{S}_i^-}^* \mathbf{S}_i^- \propto \mathcal{F}_{\mathbf{S}_i^-}(-\mathbf{S}_i^-)$ since the variables in $\mathbf{S}_i^+$ are the same as those in some other $-\mathbf{S}_i^-$.

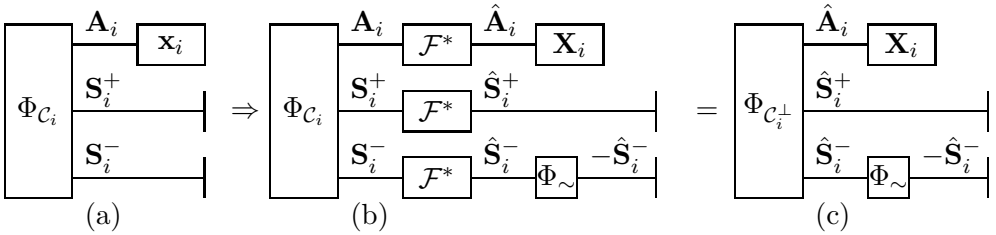

Figure 18: Dualizing local exact weight generating functions.

In particular, with a conventional state realization with constraint codes $\mathcal{C}_k$, with the convention that $\hat{S}_k$ is involved in $\mathcal{C}_k^\perp$ with a positive sign and $\hat{S}_{k+1}$ is involved in $\mathcal{C}_k^\perp$ with a negative sign, we have the following MacWilliams identity, which shows how the EWAM $\hat{\Lambda}_k(\mathbf{X}_k)$ of $\mathcal{C}_k^\perp$ may be obtained from the EWAM $\Lambda_k(\mathbf{x}_k)$ of $\mathcal{C}_k$:

$$\hat{\Lambda}_k^E(\mathbf{X}_k) \propto (\mathcal{F}_{S_k}^*)^T \Lambda_k^E(\mathcal{F}_{A_k}^* \mathbf{X}_k)(\mathcal{F}_{S_{k+1}})^T.$$

(Here the transposes are needed to conform to standard matrix conventions, even though $\mathcal{F} = \mathcal{F}^T$.)

Example 3 (cont.). Continuing with the rate-$\frac{1}{2}$ binary convolutional code of Example 3, each constraint code $\mathcal{C}_k$ has the exact weight adjacency matrix below, where we write $x_{00}, x_{01}, x_{10}, x_{11}$ for $x_k(00), x_k(01), x_k(10), x_k(11)$, respectively.

$\Lambda_k^E(\mathbf{x}_k) \quad =$

s_k/s_{k+1}	00	10	01	11
00	x_{00}	x_{11}	0	0
10	0	0	x_{01}	x_{10}
01	x_{11}	x_{00}	0	0
11	0	0	x_{10}	x_{01}

Given this EWAM for $\mathcal{C}_k$, the EWAM $\hat{\Lambda}_k^E(\mathbf{X}_k)$ of the orthogonal constraint code $\mathcal{C}_k^\perp$ is given by the matrix equation below, where we have substituted the dual indeterminates X_{00}, X_{10}, X_{01} and X_{11} for $x_{00} + x_{10} + x_{01} + x_{11}, x_{00} - x_{10} + x_{01} - x_{11}, x_{00} + x_{10} - x_{01} - x_{11}$ and $x_{00} - x_{10} - x_{01} + x_{11}$, respectively. (We have also inserted the correct scale factor.)

$$\frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} x_{00} & x_{11} & 0 & 0 \\ 0 & 0 & x_{01} & x_{10} \\ x_{11} & x_{00} & 0 & 0 \\ 0 & 0 & x_{10} & x_{01} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} = \begin{bmatrix} X_{00} & 0 & X_{11} & 0 \\ X_{11} & 0 & X_{00} & 0 \\ 0 & X_{10} & 0 & X_{01} \\ 0 & X_{01} & 0 & X_{10} \end{bmatrix}$$

From Figure 16(b), we see that this matrix is indeed the EWAM of the constraint code $\mathcal{C}_k^\perp$. □

Of course it is no surprise that a dual constraint code $\mathcal{C}_i^\perp$ is determined by $\mathcal{C}_i$, or *vice versa*; what the MacWilliams identities give us is a way of *calculating* one from the other, by taking transforms.

4.2 MacWilliams identities for other weight generating functions

Commonly each symbol alphabet A_k is equal to $\mathbb{F}^{n_k}$, the set of n_k-tuples over the ground field $\mathbb{F}$; *i.e.,* $a_k = \{a_{k\ell} \in \mathbb{F} : 1 \le \ell \le n_k\}$. In this case a transform over A_k may be expressed by n_k transforms over $\mathbb{F}$ applied to the n_k components $a_{k\ell}$ of A_k.

We may then be interested in the *complete weight generating function* (CWGF) of $\mathcal{C}$, defined as

$$g_{\mathcal{C}}^{C}(\mathbf{x}) = \sum_{\mathbf{a}\in\mathcal{A}} \Phi_{\mathcal{C}}(\mathbf{a}) \left(\prod_{k\in\mathcal{I}_{\mathcal{A}}} \prod_{\ell=1}^{n_k} x(a_{k\ell}) \right),$$

where $\mathbf{x} = \{x(a) : a \in \mathbb{F}\}$ is a set of indeterminates defined on $\mathbb{F}$. In other words, the CWGF is obtained by substituting the product $\prod_{\ell=1}^{n_k} x(a_{k\ell})$ for $x_k(a_k)$ in the EWGF.

Similarly, the *local complete weight generating function* $g_{\mathcal{C}_i}^{C}(\mathbf{x}_i, \mathbf{s}_i)$ is

$$g_{\mathcal{C}_i}^{C}(\mathbf{x}, \mathbf{s}_i) = \sum_{\mathbf{a}_i\in\mathbf{A}_i} \Phi_{\mathcal{C}_i}(\mathbf{a}_i, \mathbf{s}_i) \left(\prod_{k\in\mathcal{I}_{\mathbf{A}_i}} \prod_{\ell=1}^{n_k} x(a_{k\ell}) \right),$$

where $\mathcal{I}_{\mathbf{A}_i} \subseteq \mathcal{I}_{\mathcal{A}}$ denotes the subset of indices of symbol alphabets A_k that are involved in $\mathcal{C}_i$.

By a development parallel to that above, we have the following MacWilliams identity, which shows how the local CWGF of $\mathcal{C}_i^{\perp}$ may be computed from that of $\mathcal{C}_i$:

$$g_{\mathcal{C}_i^{\perp}}^{C}(\mathbf{X}, \hat{\mathbf{S}}_i^{+}, \hat{\mathbf{S}}_i^{-}) \propto g_{\mathcal{C}_i}^{C}(\mathcal{F}_{\mathbb{F}}^{*}\mathbf{X}, \mathcal{F}_{\mathbf{S}_i^{+}}^{*}\mathbf{S}_i^{+}, \mathcal{F}_{\mathbf{S}_i^{-}}(-\mathbf{S}_i^{-})),$$

where $\mathbf{X} = \mathcal{F}_{\mathbb{F}}\mathbf{x}$ is the dual set of indeterminates to the set $\mathbf{x}$, with $\mathcal{F}_{\mathbb{F}}$ the transform matrix on $\mathbb{F}$.

In particular, for conventional state realizations, the following MacWilliams identity shows how the complete weight adjacency matrix (CWAM) $\hat{\Lambda}_k^{C}(\mathbf{X})$ of $\mathcal{C}_k^{\perp}$ may be obtained from the CWAM $\Lambda_k^{C}(\mathbf{x})$ of $\mathcal{C}_k$:

$$\hat{\Lambda}_k^{C}(\mathbf{X}_k) \propto (\mathcal{F}_{S_k}^{*})^{T} \Lambda_k^{C}(\mathcal{F}_{\mathbb{F}}^{*}\mathbf{X})(\mathcal{F}_{S_{k+1}})^{T}.$$

Example 3 (cont.). For the rate-$\frac{1}{2}$ binary convolutional code of Example 3, each constraint code $\mathcal{C}_k$ has the CWAM

$\Lambda_k^{C}(\mathbf{x})$ =

s_k/s_{k+1}	00	10	01	11
00	x_0^2	x_1^2	0	0
10	0	0	x_0x_1	x_0x_1
01	x_1^2	x_0^2	0	0
11	0	0	x_0x_1	x_0x_1

where we have written x_0 and x_1 instead of $x(0)$ and $x(1)$, respectively. The CWAM $\hat{\Lambda}_k^{C}(\mathbf{X})$ of the orthogonal constraint code $\mathcal{C}_k^{\perp}$ is given by the matrix equation below, where we have substituted the dual indeterminates X_0 and X_1 for $x_0 + x_1$ and $x_0 - x_1$, respectively.

$$\frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} x_0^2 & x_1^2 & 0 & 0 \\ 0 & 0 & x_0x_1 & x_0x_1 \\ x_1^2 & x_0^2 & 0 & 0 \\ 0 & 0 & x_0x_1 & x_0x_1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} = \begin{bmatrix} X_0^2 & 0 & X_1^2 & 0 \\ X_1^2 & 0 & X_0^2 & 0 \\ 0 & X_0X_1 & 0 & X_0X_1 \\ 0 & X_0X_1 & 0 & X_0X_1 \end{bmatrix}$$

We see from Figure 16(b) that this matrix is indeed the CWAM of $\mathcal{C}_k^{\perp}$. □

Here the point is that even though the CWGF of $\mathcal{C}_i$ does not fully determine $\mathcal{C}_i$, it does determine the CWGF of $\mathcal{C}_i^{\perp}$, and *vice versa*.

The *Hamming weight generating function* (HWGF) $g_{\mathcal{C}}^{H}(x)$ of a linear code $\mathcal{C}$ may be obtained by substituting 1 for $x(0)$ and x for each $x(a), a \neq 0$, in $g_{\mathcal{C}}^{C}(\mathbf{x})$, and similarly for the *Hamming*

weight adjacency matrix (HWAM) $\Lambda_k^H(x)$ of a constraint code $\mathcal{C}_k$ in a conventional state realization. Thus each element of $\Lambda_k^H(x)$ becomes a polynomial of degree n_k or less in the single indeterminate x. The dual indeterminates become $X(0) = 1 + (|\mathbb{F}| - 1)x$ and $X(\hat{a}) = 1 - x$ for $\hat{a} \neq 0$, which scale to 1 and $X = (1 - x)/(1 + (|\mathbb{F}| - 1)x)$, respectively. Substituting in the above MacWilliams identities for CWGFs or CWAMs, we obtain MacWilliams identities for HWGFs or HWAMs. This yields the main result of [9, 10].[6]

Example 3 (cont.). For the rate-1/2 binary convolutional code $\mathcal{C}$ of Example 3, each constraint code $\mathcal{C}_k$ has the HWAM

$$\Lambda^H(x) \quad = \quad \begin{array}{|c|cccc|} \hline s/s' & 00 & 10 & 01 & 11 \\ \hline 00 & 1 & x^2 & 0 & 0 \\ 10 & 0 & 0 & x & x \\ 01 & x^2 & 1 & 0 & 0 \\ 11 & 0 & 0 & x & x \\ \hline \end{array}$$

For the orthogonal code $\mathcal{C}^\perp$, each constraint code $\mathcal{C}_k^\perp$ has the HWAM

$$\hat{\Lambda}^H(X) \quad = \quad \begin{array}{|c|cccc|} \hline s/s' & 00 & 10 & 01 & 11 \\ \hline 00 & 1 & 0 & X^2 & 0 \\ 10 & X^2 & 0 & 1 & 0 \\ 01 & 0 & X & 0 & X \\ 11 & 0 & X & 0 & X \\ \hline \end{array}$$

The reader may verify that $\Lambda^H(x)$ and $\hat{\Lambda}^H(X)$ satisfy a MacWilliams identity with $X = \frac{1-x}{1+x}$. [Hint: it may be easier to start with CWAMs.] Note that here $\hat{\Lambda}^H(x)$ happens to be the transpose of $\Lambda^H(x)$. □

Example 4 (cont.). For a worked-out example of the HWAM $\hat{\Lambda}_k^H(X)$ of the orthogonal code $\mathcal{C}_k^\perp$ to the constraint code $\mathcal{C}_k$ of Example 4, see [10]. □

Again, the point is that even though the HWGF of $\mathcal{C}_i$ does not fully determine $\mathcal{C}_i$, it does determine the HWGF of $\mathcal{C}_i^\perp$, and *vice versa*.

Ericson and Zinoviev [3] have generalized this method of obtaining further weight generating functions from complete weight generating functions, as follows. A partition of a finite abelian group G into disjoint subsets $\{G_j\}$ and of its character group $\hat{G}$ into disjoint subsets $\{\hat{G}_i\}$ is called a *Fourier-invariant pair* if for all i, j the transform of the indicator function $\Phi(G_j)$ of G_j, namely

$$\hat{\Phi}(\hat{g}) = \sum_{g \in G} \Phi(G_j)\langle \hat{g}, g \rangle = \sum_{g \in G_j} \langle \hat{g}, g \rangle,$$

depends only on the subset $\hat{G}_i$ that contains $\hat{g}$, and similarly for the inverse transform, where $\langle \hat{g}, g \rangle = \hat{g}(g)$. For example, the Hamming partition pair defined by $G_0 = \{0\}, G_1 = G \setminus \{0\}$ and $\hat{G}_0 = \{0\}, \hat{G}_1 = \hat{G} \setminus \{0\}$ is Fourier-invariant. For any Fourier-invariant partition pair, a MacWilliams identity may be obtained for the corresponding generalized weight generating function; see [3] or [4]. Zinoviev and Ericson [17] show that this concept is equivalent to that of an association scheme.

[6]The MacWilliams identity of [9, 10] is stated in terms of the HWAM for a minimal realization of a linear time-invariant convolutional code $\mathcal{C}$ in controller canonical form, and the HWAM of *some* minimal encoder for the orthogonal code $\mathcal{C}^\perp$. Our results apply to the CWAM or HWAM of any state realization, and the CWAM or HWAM of its dual realization, because in our development, by constraint code duality, the basis of the dual state space representation is fixed as soon as the basis of the primal state space is fixed.

5 MacWilliams identities for terminated convolutional codes

A principal use of MacWilliams identities is to compute the distance distribution of a linear code $\mathcal{C}$ from that of its dual code $\mathcal{C}^\perp$, where typically $\mathcal{C}$ is high-rate and $\mathcal{C}^\perp$ is low-rate (*i.e.,* $|\mathcal{C}^\perp| < |\mathcal{C}|$).

For a convolutional code $\mathcal{C}$, the most commonly studied distance distribution is its *free (Hamming) distance spectrum*, namely, the distribution of (Hamming) weights of codewords in $\mathcal{C}$ that start and end in the zero state without passing through an intermediate zero state.[7] Shearer and McEliece [16] showed by example that the free distance spectrum of $\mathcal{C}$ does not in general determine that of $\mathcal{C}^\perp$, and therefore that there could be no MacWilliams identity for such distributions.

Recently, Bocharova, Hug, Johannesson and Kudryashov [2] have proved a MacWilliams identity for truncations of a convolutional code $\mathcal{C}$ and its orthogonal code $\mathcal{C}^\perp$. By letting the truncation length become large, they obtain an approximation to the free distance spectrum of $\mathcal{C}$.

In this section, we derive similar results for weight distributions of codes obtained by various kinds of termination procedures, of which we regard tail-biting as the most elegant. We argue that these alternative distributions are as useful for estimating code performance as the free distance spectrum. These results effectively answer the original question posed by Shearer and McEliece [16], which we would state as follows: is there a duality relationship that allows us to estimate the performance parameters of $\mathcal{C}^\perp$ from those of $\mathcal{C}$?

5.1 Terminated convolutional codes

A general method for approximating the free distance spectrum of a linear convolutional code $\mathcal{C}$ is to derive a series of block codes $\mathcal{C}_N$ of length N from $\mathcal{C}$ by some sort of termination procedure, and then to study the distance distributions of $\mathcal{C}_N$ as $N \to \infty$. As we will see, for any of the termination methods below, the distance distribution of $\mathcal{C}_N$, normalized by N, approaches the free distance spectrum of $\mathcal{C}$ for $d_{\rm free} \le d < 2d_{\rm free}$, where $d_{\rm free}$ denotes the free distance of $\mathcal{C}$ (the least weight of any nonzero code sequence). However, we will argue that tail-biting is the nicest, particularly if we are also interested in the distance distribution of the orthogonal convolutional code $\mathcal{C}^\perp$.

The left side of Figure 19 shows normal graphs of five block codes $\mathcal{C}_N$ obtained from a convolutional code $\mathcal{C}$ by five kinds of termination procedures:

(a) the subcode $\mathcal{C}_{[0,N)}$;

(b) the projection $\mathcal{C}_{|[0,N)}$;

(c) the truncated code $\mathcal{C}_{\triangleleft[0,N)}$;

(d) the reverse-truncated code $\mathcal{C}_{\triangleright[0,N)}$; and

(e) the tail-biting code $\mathcal{C}_{||[0,N)}$.

In each case the central part of the graph consists of N consecutive trellis sections of $\mathcal{C}$, and the block code symbols are the corresponding convolutional code symbols $(a_0, \ldots, a_{n-1})$.

To obtain the subcode $\mathcal{C}_{[0,N)}$, the starting and ending state variables are constrained to be zero: $s_0 = s_N = 0$. For the projection $\mathcal{C}_{|[0,N)}$, the starting and ending states may be any arbitrary pair

[7] We will assume in this section that the unique state sequence associated with the all-zero code sequence is the all-zero state sequence; this assumption can always be satisfied by choosing a minimal encoder for $\mathcal{C}$.

$s_0 \in S_0, s_N \in S_N$. For the truncated code $\mathcal{C}_{\lhd[0,N)}$, we constrain $s_0 = 0$, but let s_N be arbitrary; for the reverse-truncated code, the reverse constraints are imposed. Finally, for the tail-biting code $\mathcal{C}_{||[0,N)}$, we impose the constraint $s_N = s_0$.

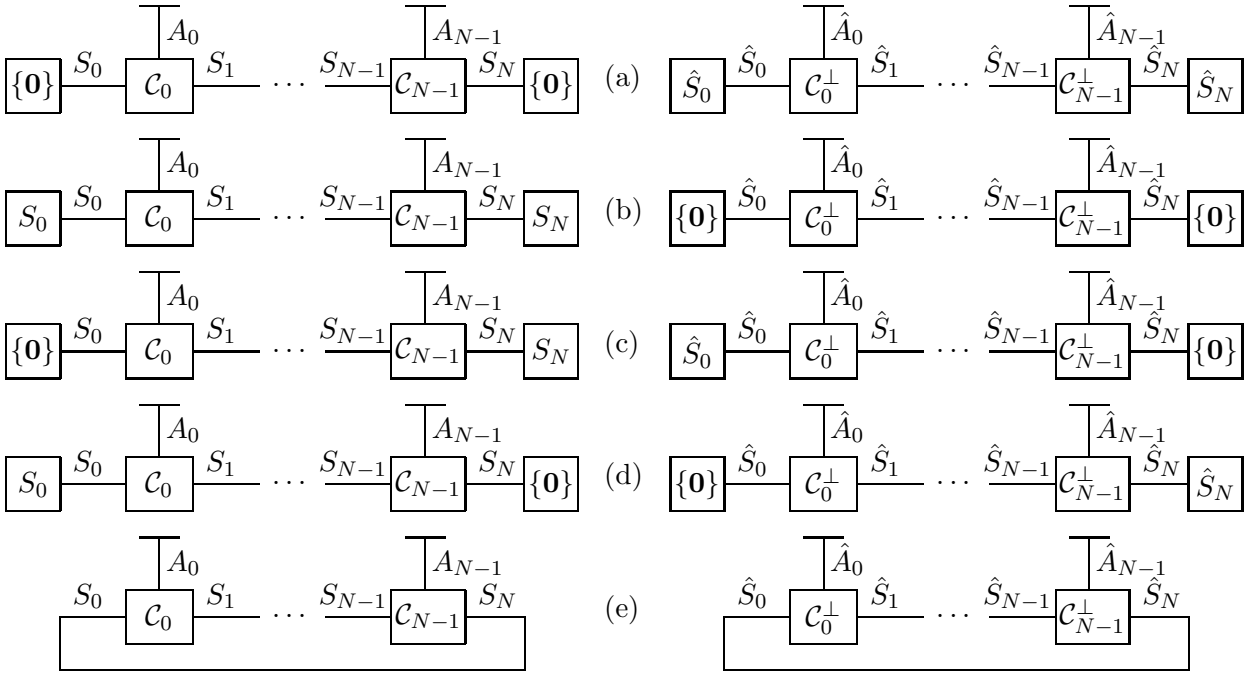

Figure 19: Terminated convolutional codes and their duals (over a field of characteristic 2).

The right side of Figure 19 shows the orthogonal codes to all of these codes, obtained simply by use of the normal graph duality theorem, along with the observation that the orthogonal code to a trivial code $\{\mathbf{0}\}$ is a universe code, *e.g.,* $\hat{S}_0$, and *vice versa*. (For simplicity, we have assumed that the characteristic of $\mathbb{F}$ is 2, so that we do not need to show sign inverters.)

We observe that the orthogonal code to a subcode $\mathcal{C}_{[0,N)}$ of $\mathcal{C}$ is the corresponding projection $(\mathcal{C}^\perp)_{|[0,N)}$ of $\mathcal{C}^\perp$, and *vice versa*, as is well known; the orthogonal code to a truncated code $\mathcal{C}_{\lhd[0,N)}$ of $\mathcal{C}$ is the corresponding reverse-truncated code $(\mathcal{C}^\perp)_{\rhd[0,N)}$ of $\mathcal{C}^\perp$, and *vice versa* [2]; and the orthogonal code to a tail-biting code $\mathcal{C}_{||[0,N)}$ of $\mathcal{C}$ is the corresponding tail-biting code $(\mathcal{C}^\perp)_{||[0,N)}$ of $\mathcal{C}^\perp$ [5].

Since $\mathcal{C}_{[0,N)}$ and $(\mathcal{C}^\perp)_{|[0,N)}$, or $\mathcal{C}_{\lhd[0,N)}$ and $(\mathcal{C}^\perp)_{\rhd[0,N)}$, or $\mathcal{C}_{||[0,N)}$ and $(\mathcal{C}^\perp)_{||[0,N)}$ are orthogonal block codes, there is a MacWilliams identity between their respective weight generating functions.

We now give examples that will illustrate these various kinds of terminated codes, and the following general properties:

- The rate of a subcode $\mathcal{C}_{[0,N)}$ is less than that of $\mathcal{C}$, the rate of a projection is higher than that of $\mathcal{C}$, and the rate of a truncated or a tail-biting code is the same as that of $\mathcal{C}$.

- The minimum distance of a subcode $\mathcal{C}_{[0,N)}$ is (at least) the same as the minimum free distance d_{free} of $\mathcal{C}$. As is well known, a tail-biting code has the same property, provided that N is large enough. However, the other kinds of terminated codes in general have low-weight codewords.

Example 3 (cont.). We now consider various methods of terminating the rate-$\frac{1}{2}$ binary convolutional code $\mathcal{C}$ of Example 3 with a block length of $N = 4$.

The subcode $\mathcal{C}_{[0,4)}$ is the $(8, 2)$ binary linear block code generated by the two generators

$$\begin{array}{cccc} 11 & 01 & 11 & 00 \\ 00 & 11 & 01 & 11 \end{array}$$

The minimum distance of this block code is the same as the minimum free distance $d_{\text{free}} = 5$ of $\mathcal{C}$, although its rate is lower.

The orthogonal code to the subcode $\mathcal{C}_{[0,4)}$ is the projection $(\mathcal{C}^{\perp})_{|[0,4)}$ of the orthogonal convolutional code $\mathcal{C}^{\perp}$, which is the $(8, 6)$ binary linear block code generated by the six generators

$$\begin{array}{cccc} 11 & 00 & 00 & 00 \\ 10 & 11 & 00 & 00 \\ 11 & 10 & 11 & 00 \\ 00 & 11 & 10 & 11 \\ 00 & 00 & 11 & 10 \\ 00 & 00 & 00 & 11 \end{array}$$

The minimum distance of this block code is 2, less than the minimum free distance of $\mathcal{C}^{\perp}$, although its rate is higher.

The truncated code $\mathcal{C}_{\lhd[0,4)}$ is the $(8, 4)$ binary linear block code generated by

$$\begin{array}{cccc} 11 & 01 & 11 & 00 \\ 00 & 11 & 01 & 11 \\ 00 & 00 & 11 & 01 \\ 00 & 00 & 00 & 11 \end{array}$$

The minimum distance of this block code is 2, but its rate is the same as that of $\mathcal{C}$. Its orthogonal code $(\mathcal{C}^{\perp})_{\rhd[0,4)}$ is the $(8, 4)$ binary linear block code generated by

$$\begin{array}{cccc} 11 & 00 & 00 & 00 \\ 10 & 11 & 00 & 00 \\ 11 & 10 & 11 & 00 \\ 00 & 11 & 10 & 11 \end{array}$$

which has the same parameters.

The tail-biting terminated code $\mathcal{C}_{||[0,4)}$ is the $(8, 4)$ binary linear block code generated by

$$\begin{array}{cccc} 11 & 01 & 11 & 00 \\ 00 & 11 & 01 & 11 \\ 11 & 00 & 11 & 01 \\ 01 & 11 & 00 & 11 \end{array}$$

whereas the orthogonal tail-biting terminated code $(\mathcal{C}^{\perp})_{||[0,N)}$ is the $(8, 4)$ binary linear block code generated by the four generators

$$\begin{array}{cccc} 11 & 10 & 11 & 00 \\ 00 & 11 & 10 & 11 \\ 11 & 00 & 11 & 10 \\ 10 & 11 & 00 & 11 \end{array}$$

Both of these codes have a minimum distance of only 2 (*e.g.,* for paths such as 01 00 01 00 from state 10 to state 10 in $\mathcal{C}_{||[0,4)}$). However, for $N \geq 10$, it turns out that the minimum distance of both tail-biting terminated codes is 5, the same as the minimum free distance of $\mathcal{C}$ or $\mathcal{C}^{\perp}$. □

5.2 Distance distributions of convolutional codes and terminated codes

In this subsection, we consider how the free distance spectrum of a linear time-invariant convolutional code $\mathcal{C}$ may be derived from the weight distribution of any of these terminated codes of length N as $N \to \infty$. Then, in the next subsection, we show how the weight generating functions of any of these terminated codes may be determined from the weight adjacency matrix of the convolutional code. Again, the most elegant relationships are obtained for tail-biting terminated codes.

We continue to assume that the unique state sequence associated with the infinite all-zero code sequence is the all-zero state sequence. Consequently, the lowest-weight words of a terminated code as $N \to \infty$ must be those that pass through the zero state almost all of the time. These code sequences are as follows, for the various termination methods we have considered:

- If we terminate to the subcode $\mathcal{C}_{[0,N)}$, then code sequences start and end in the zero state, and the lowest-weight sequences correspond to the lowest-weight sequences in the free distance spectrum. If the minimum free distance is d_{free}, then for $d_{\text{free}} \leq d < 2d_{\text{free}}$ there will be approximately $N \times N_d$ sequences in the terminated code of weight d, where N_d is the number of code sequences of weight d in the free distance spectrum of $\mathcal{C}$. Thus, for $d_{\text{free}} \leq d < 2d_{\text{free}}$, the weight distribution per unit time of $\mathcal{C}$ is the limit of the weight distribution of $\mathcal{C}_{[0,N)}$ normalized by (divided by) N as $N \to \infty$. For $d \geq 2d_{\text{free}}$, there will be overcounting— *e.g.,* two sequences of weight d_{free} may be counted as one of weight $2d_{\text{free}}$— but we will argue below that such overcounting should not affect estimates of code performance.

- If we terminate to the projection $\mathcal{C}_{|[0,N)}$, then code sequences can start and end in any state, and there will be low-weight sequences starting with a low-weight state transition $s \to 0$, remaining in state 0 for nearly N time units, and then ending with a low-weight transition $0 \to s'$, where s and s' are not both 0. Thus the minimum distance of $\mathcal{C}_{|[0,N)}$ will be less than d_{free} for all N. However, the number of such low-weight sequences remains constant, so after normalization we will eventually see the same normalized weight distribution as for $\mathcal{C}_{[0,N)}$.

- If we terminate to the truncated code $\mathcal{C}_{\lhd[0,N)}$, then by the same argument we will eventually see the correct normalized weight distribution. In this case, for a code sequence that starts in the zero state, remains there for nearly N time units, and then ends with a low-weight transition $0 \to s$, the total weight is only that of the low-weight transition $0 \to s$. However, again the number of such low-weight sequences remains constant, so after normalization we will eventually see the correct normalized weight distribution.

- If we terminate to the tail-biting code $\mathcal{C}_{||[0,N)}$, then by the same argument we will eventually see the correct normalized weight distribution. Note however that in this case the total weight of a code sequence starting with a low-weight transition $s \to 0$, remaining in the zero state for nearly N time units, and then ending with a low-weight transition $0 \to s$, must be at least d_{free}, since the ending sequence (corresponding to the state transition $0 \to s$) followed by the starting sequence (corresponding to $s \to 0$) must be a cyclic shift of a code sequence of $\mathcal{C}$. Thus the minimum distance of $\mathcal{C}_{||[0,N)}$ must equal d_{free} for large enough N.

We conclude that as $N \to \infty$ the normalized weight distribution of any of these terminated codes approaches the free distance spectrum of $\mathcal{C}$ for $d_{\text{free}} \leq d < 2d_{\text{free}}$. However, only the tail-biting code has the same rate as $\mathcal{C}$ and the same minimum distance d_{free} (for large enough N).

We now argue that the normalized weight distribution of any of these terminated codes $\mathcal{C}_N$ must yield the same estimate of code performance over N time units as the free distance spectrum of $\mathcal{C}$, if these estimates are accurate. The probability of error event $P(\mathcal{E})$ of $\mathcal{C}$ per unit time may be estimated using the free distance spectrum. The probability of any error in N time units is then estimated as $NP(\mathcal{E})$. If this is a good estimate (implying $N < 1/P(\mathcal{E})$), then the probability of two or more error events in N time units must be negligible. But the probability of any error in decoding $\mathcal{C}$ over N time units is essentially the same as the probability of block decoding error in decoding $\mathcal{C}_N$, which may be estimated by the weight distribution of N, which counts codewords that include two or more error events. If the probability of two or more error events in N time units is negligible, then an estimate based on the weight distribution of $\mathcal{C}_N$ must approximately agree with an estimate based on the free distance spectrum of $\mathcal{C}$.

5.3 Free distance spectra for convolutional codes from terminated codes

We now show how weight generating functions for terminations of a linear time-invariant convolutional code $\mathcal{C}$ may be derived from the weight adjacency matrix of the constraint code $\mathcal{C}_k$ that specifies $\mathcal{C}$. This will allow us to state MacWilliams identities for terminated convolutional codes, and to estimate code performance.

We compute the Hamming weight distributions of these terminated codes as follows. Let $\Lambda_{[0,N)}(x)$ be the Hamming weight adjacency matrix of $\mathcal{C}$ over the interval $[0, N)$, whose elements are indexed by $S_0 \times S_N$. As we have seen in Example 2, if $\mathcal{C}$ is time-invariant and $\Lambda(x)$ is the HWAM of each constraint code $\mathcal{C}_k$, then $\Lambda_{[0,N)}(x)$ is simply equal to $\Lambda^N(x)$.

From their definitions, we see that the HWGFs of various terminated codes of $\mathcal{C}$ can be read as follows from the HWAM $\Lambda^N(x)$:

(a) The HWGF of the subcode $\mathcal{C}_{[0,N)}$ is the $(0, 0)$ element of $\Lambda^N(x)$.

(b) The HWGF of the projection $\mathcal{C}_{|[0,N)}$ is the sum of all elements of $\Lambda^N(x)$.

(c) The HWGF of the truncated code $\mathcal{C}_{\lhd[0,N)}$ is the sum of all elements in the first row of $\Lambda^N(x)$.

(d) The HWGF of the reverse-truncated code $\mathcal{C}_{\rhd[0,N)}$ is the sum of all elements in the first column of $\Lambda^N(x)$.

(e) The HWGF of the tail-biting code $\mathcal{C}_{||[0,N)}$ is the sum of all diagonal elements of $\Lambda^N(x)$; *i.e.,* its trace Tr $(\Lambda^N(x))$ (see Example 5).

Example 3 (cont.). For the rate-1/2 binary convolutional code $\mathcal{C}$ of Example 3, the HWAM of a section consisting of $N = 2$ time units of our example code $\mathcal{C}$ is thus

$$\Lambda^2(x) \quad = \quad \begin{bmatrix} 1 & x^2 & x^3 & x^3 \\ x^3 & x & x^2 & x^2 \\ x^2 & x^4 & x & x \\ x^3 & x & x^2 & x^2 \end{bmatrix}.$$

This shows that there is exactly one path from each state in S_k to each state in S_{k+2}, and that the minimum Hamming weight of any of these paths (other than the zero path) is 1.

For a section consisting of $N = 4$ time units of this code, the HWAM is

$$\Lambda^4(x) = \begin{bmatrix} 1+2x^5+x^6 & x^2+x^3+x^4+x^7 & x^3+2x^4+x^5 & x^3+2x^4+x^5 \\ x^3+2x^4+x^5 & x^2+x^3+x^5+x^6 & 2x^3+x^4+x^6 & 2x^3+x^4+x^6 \\ x^2+x^3+x^4+x^7 & x^2+x^4+2x^5 & x^2+x^3+x^5+x^6 & x^2+x^3+x^5+x^6 \\ x^3+2x^4+x^5 & x^2+x^3+x^5+x^6 & 2x^3+x^4+x^6 & 2x^3+x^4+x^6 \end{bmatrix}.$$

This shows that there are four paths from each state in S_k to each state in S_{k+4}, and that the minimum nonzero Hamming weight of any of these paths is 2.

The Hamming weight generating function of the tail-biting termination $\mathcal{C}_{|||[0,4)}$ of length 4 is the trace of $\Lambda^4(x)$, namely $1 + 2x^2 + 4x^3 + x^4 + 4x^5 + 4x^6$. Since $\hat{\Lambda}^4(x)$ happens to be the transpose of $\Lambda^4(x)$, the orthogonal tail-biting terminated code $(\mathcal{C}^\perp)_{|||[0,4)}$ has the same Hamming weight generating function. It is easy to check that the Hamming weight generating function of this code is indeed invariant under the MacWilliams transform. □

Using tail-biting terminated codes, and normalizing the weight distribution by dividing by N, we have that the generating function of the normalized Hamming weight distribution of $\mathcal{C}$ is

$$g_{\mathcal{C}}(x) = \lim_{N\to\infty} \frac{1}{N} \mathrm{Tr}\ (\Lambda^N(x)).$$

Moreover, there is a MacWilliams identity between $g_{\mathcal{C}}(x)$ and $g_{\mathcal{C}^\perp}(x)$. The performance of $\mathcal{C}$ may be estimated from $g_{\mathcal{C}}(x)$, and that of $\mathcal{C}^\perp$ from $g_{\mathcal{C}^\perp}(x)$. (Similar observations are made in [2], using truncated codes.)

It appears that the behavior of $g_{\mathcal{C}}(x)$ might be analyzed by using an extension of Perron-Frobenius theory to generating function matrices, as in [8]; however, we have not attempted such an analysis.

Example 1 (cont.). For a section consisting of $N = 16$ time units of the rate-1/2 binary convolutional code $\mathcal{C}$ of Example 1, the HWAM $\Lambda^{16}(x)$ (modulo x^8) is

$$\begin{bmatrix} 1+14x^5+25x^6+44x^7 & x^2+x^3+2x^4+4x^5+8x^6+29x^7 & x^3+2x^4+4x^5+8x^6+16x^7 & x^3+2x^4+4x^5+8x^6+16x^7 \\ x^3+2x^4+4x^5+8x^6+16x^7 & x^5+3x^6+8x^7 & x^6+4x^7 & x^6+4x^7 \\ x^2+x^3+2x^4+4x^5+8x^6+29x^7 & x^4+2x^5+5x^6+12x^7 & x^5+3x^6+8x^7 & x^5+3x^6+8x^7 \\ x^3+2x^4+4x^5+8x^6+16x^7 & x^5+3x^6+8x^7 & x^6+4x^7 & x^6+4x^7 \end{bmatrix}$$

Notice that

$$\mathrm{Tr}\ (\Lambda^{16}(x)) = 1 + 16x^5 + 32x^6 + 64x^7 + \cdots,$$

so that normalizing the distribution by dividing the higher-order coefficients by $N = 16$ already gives the precise free distance spectrum of $\mathcal{C}$ for $d < 8$, namely $x^5 + 2x^6 + 4x^7 + \cdots$. Thus the convergence to the limiting generating function $g_{\mathcal{C}}(x)$ is rapid and exact. This property of tail-biting codes is not shared by other kinds of terminations. □

Example 6 (*cf.* [16, 2]). The two codes proposed by Shearer and McEliece [16] for their counterexample provide an excellent final example. The first code is a rate-1/3 binary linear time-invariant convolutional code $\mathcal{C}_1$ generated by the degree-1 generators $(1, 1+D, D)$, *i.e.,* $\mathcal{C}_1$ is generated by a minimal encoder with impulse response is $(110, 011, 000, \ldots)$, whose trellis section is shown in Figure 20(a). The HWAM of this encoder is

$$\Lambda_1(x) \quad = \quad \begin{bmatrix} 1 & x^2 \\ x^2 & x^2 \end{bmatrix}.$$

(a) (b)

Figure 20: Trellis sections of (a) rate-1/3 2-state binary convolutional code $\mathcal{C}_1$; (b) similar code $\mathcal{C}_2$.

The second code is a rate-1/3 binary linear time-invariant convolutional code $\mathcal{C}_2$ generated by the degree-1 generators $(D, D, 1+D)$, *i.e.,* $\mathcal{C}_2$ is generated by a minimal encoder whose impulse response is $(001, 111, 000, \ldots)$, whose trellis section is shown in Figure 20(b). The HWAM of this encoder is

$$\Lambda_2(x) \quad = \quad \begin{bmatrix} 1 & x \\ x^3 & x^2 \end{bmatrix}.$$

Since the weights of the $0 \to 0$ and $1 \to 1$ transitions are the same for $\mathcal{C}_1$ and $\mathcal{C}_2$, and since the sums of the weights of the $0 \to 1$ and $1 \to 0$ transitions are the same, it is evident that the weight distributions of the subcodes $(\mathcal{C}_1)_{[0,N)}$ and $(\mathcal{C}_2)_{[0,N)}$ are the same for all N, and that the free distance spectra of $\mathcal{C}_1$ and $\mathcal{C}_2$ are also the same. For the same reason, the weight distributions of the tail-biting codes $(\mathcal{C}_1)_{|||[0,N)}$ and $(\mathcal{C}_2)_{|||[0,N)}$ are the same for all N.

However, the weight distributions of the projections $(\mathcal{C}_1)_{|[0,N)}$ and $(\mathcal{C}_2)_{|[0,N)}$ are not the same even for $N = 1$. It follows that the weight distributions of the subcodes $(\mathcal{C}_1^\perp)_{[0,N)}$ and $(\mathcal{C}_2^\perp)_{[0,N)}$ of their orthogonal rate-2/3 codes $\mathcal{C}_1^\perp$ and $\mathcal{C}_2^\perp$ are not the same, and therefore that their free distance spectra are not the same; this was the point of Shearer and McEliece [16].

On the other hand, since the weight distributions of the tail-biting codes $(\mathcal{C}_1)_{|||[0,N)}$ and $(\mathcal{C}_2)_{|||[0,N)}$ are the same for all N, it follows that *the weight distributions of the tail-biting codes* $(\mathcal{C}_1^\perp)_{|||[0,N)}$ *and* $(\mathcal{C}_2^\perp)_{|||[0,N)}$ *are the same for all* N.

Since the performance of $\mathcal{C}_1^\perp$ and $\mathcal{C}_2^\perp$ may be estimated from these weight distributions as $N \to \infty$, it follows that the performance of $\mathcal{C}_1^\perp$ and $\mathcal{C}_2^\perp$ is effectively the same, despite the difference in their free distance spectra.[8] □

In summary, similarly to [2], we have shown that there is a MacWilliams identity between the generating functions of the weight distributions per unit time of a linear convolutional code $\mathcal{C}$ and its orthogonal code $\mathcal{C}^\perp$ (as calculated from their tail-biting terminations), and that these distributions are as useful as their free distance spectra for estimating code performance. These results effectively answer the question posed by Shearer and McEliece [16].

[8] Another way of reaching the same (or a stronger) conclusion is to observe that $\mathcal{C}_1$ and $\mathcal{C}_2$ are equivalent under a simple finite-memory permutation. Therefore $\mathcal{C}_1^\perp$ and $\mathcal{C}_2^\perp$ must be equivalent under the same permutation, and thus must have precisely the same performance on a memoryless channel with maximum likelihood decoding.

Acknowledgments

For discussions that led to the conceptual framework of partition functions of normal factor graphs, I am deeply grateful to Pascal Vontobel and Yongyi Mao. I also thank Heide Gluesing-Luerssen, Rolf Johannesson, Pascal Vontobel and the reviewers for many helpful comments on earlier versions of this paper.

References

[1] A. Al-Bashabsheh and Y. Mao, "Normal factor graphs and holographic transformations," to appear, *IEEE Trans. Inform. Theory*, Feb. 2011. ArXiv: 1004.3833 [cs.IT].

[2] I. E. Bocharova, F. Hug, R. Johannesson and B. Kudryashov, "On weight enumerators and MacWilliams identity for convolutional codes," *Proc. ITA Workshop* (La Jolla, CA), Feb. 2010.

[3] T. Ericson and V. A. Zinoviev, "On Fourier-invariant partitions of finite abelian groups and the MacWilliams identity for group codes," *Problemy Peredachi Informatsii*, vol. 32, pp. 137–143, 1996.

[4] G. D. Forney, Jr., "Transforms and groups," in *Codes, Curves and Signals: Common Threads in Communications* (A. Vardy, ed.), pp. 79–97. Boston: Kluwer, 1998.

[5] G. D. Forney, Jr., "Codes on graphs: Normal realizations," *IEEE Trans. Inform. Theory*, vol. 47, pp. 520–548, Feb. 2001.

[6] G. D. Forney, Jr., "MacWilliams identities for codes on graphs," *Proc. IEEE Inform. Theory Workshop* (Taormina, Italy), pp. 120–124, Oct. 2009.

[7] G. D. Forney, Jr., "MacWilliams identities for terminated convolutional codes," *Proc. IEEE Intl. Symp. Inform. Theory* (Austin, TX), June 2010.

[8] G. D. Forney, Jr., F. R. Kschischang, B. Marcus and S. Tuncel, "Iterative decoding of tail-biting trellises and connections with symbolic dynamics," in *Codes, Systems and Graphical Models* (B. Marcus and J. Rosenthal, eds.), pp. 239–264. New York: Springer, 2001.

[9] H. Gluesing-Luerssen and G. Schneider, "On the MacWilliams identity for convolutional codes," *IEEE Trans. Inform. Theory*, vol. 54, pp. 1536–1550, April 2008.

[10] H. Gluesing-Luerssen and G. Schneider, "A MacWilliams identity for convolutional codes: The general case," *IEEE Trans. Inform. Theory*, vol. 55, pp. 2920–2930, July 2009.

[11] F. R. Kschischang, B. J. Frey and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inform. Theory*, vol. 47, pp. 498–519, Feb. 2001.

[12] H.-A. Loeliger, "An introduction to factor graphs," *IEEE Sig. Proc. Mag.*, vol. 21, pp. 28–41, Jan. 2004.

[13] H.-A. Loeliger, J. Dauwels, J. Hu, S. Korl, L. Ping and F. R. Kschischang, "The factor graph approach to model-based signal processing," *Proc. IEEE*, vol. 95, pp. 1295–1322, June 2007.

[14] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. New York: North-Holland, 1977.

[15] Y. Mao and F. R. Kschischang, "On factor graphs and the Fourier transform," *IEEE Trans. Inform. Theory*, vol. 51, pp. 1635–1649, May 2005.

[16] J. B. Shearer and R. J. McEliece, "There is no MacWilliams identity for convolutional codes," *IEEE Trans. Inform. Theory*, vol. 23, pp. 775–776, Nov. 1977.

[17] V. A. Zinoviev and T. Ericson, "Fourier-invariant pairs of partitions of finite abelian groups and association schemes," *Probs. Inform. Transm.*, vol. 45, pp. 221–231, Sept. 2009.

Author biography

G. David Forney, Jr. received the B.S.E. degree in electrical engineering from Princeton University, Princeton, NJ, in 1961, and the M.S. and Sc.D. degrees in electrical engineering from the Massachusetts Institute of Technology (M.I.T.), Cambridge, MA, in 1963 and 1965, respectively.

From 1965-99 he was with the Codex Corporation, which was acquired by Motorola, Inc. in 1977, and its successor, the Motorola Information Systems Group, Mansfield, MA. Since 1996, he has been an Adjunct Professor at M.I.T.

Dr. Forney was Editor of the IEEE Transactions on Information Theory from 1970 to 1973. He has been a member of the Board of Governors of the IEEE Information Theory Society during 1970-76, 1986-94, and 2004-10, and was President in 1992 and 2008. He has been awarded the 1970 IEEE Information Theory Group Prize Paper Award, the 1972 IEEE Browder J. Thompson Memorial Prize Paper Award, the 1990 and 2009 IEEE Donald G. Fink Prize Paper Awards, the 1992 IEEE Edison Medal, the 1995 IEEE Information Theory Society Claude E. Shannon Award, the 1996 Christopher Columbus International Communications Award, and the 1997 Marconi International Fellowship. In 1998 he received an IT Golden Jubilee Award for Technological Innovation, and two IT Golden Jubilee Paper Awards. He received an honorary doctorate from EPFL, Lausanne, Switzerland in 2007. He was elected a Fellow of the IEEE in 1973, a member of the National Academy of Engineering (U.S.A.) in 1983, a Fellow of the American Association for the Advancement of Science in 1993, an honorary member of the Popov Society (Russia) in 1994, a Fellow of the American Academy of Arts and Sciences in 1998, and a member of the National Academy of Sciences (U.S.A.) in 2003.