MacWilliams identities for codes on graphs

MacWilliams Identities for Codes on Graphs

G. David Forney, Jr.
Laboratory for Information and Decision Systems
Massachusetts Institute of Technology
Cambridge, MA 02139
Email: forneyd@comcast.net

***Abstract*—The MacWilliams identity for linear time-invariant convolutional codes that has recently been found by Gluesing-Luerssen and Schneider is proved concisely, and generalized to arbitrary group codes on graphs. A similar development yields a short, transparent proof of the dual sum-product update rule.**

I. Introduction

Finding a MacWilliams-type identity for convolutional codes is a problem of long standing. Recently Gluesing-Luerssen and Schneider (GLS) have formulated [1] and proved [2] such an identity involving the (Hamming) weight adjacency matrix (WAM) of a linear time-invariant convolutional code over a finite field and the WAM of its orthogonal code.

The purpose of this note is to provide a concise group-theoretic proof of this identity, and to generalize it to arbitrary group codes defined on graphs. We use first the general duality result that, given a "normal" graphical realization of a group code $\mathcal{C}$, the dual (orthogonal) code $\mathcal{C}^\perp$ is realized by the dual graph, in which the "constraint code" corresponding to each node is replaced by its orthogonal code [3]. A more or less standard development (following [4]), using the Poisson summation formula, then proves an appropriate MacWilliams identity between the complete or Hamming WAM of a constraint code and the complete or Hamming WAM of its dual.

In the special case of a state-space (trellis) realization of a linear time-invariant convolutional code over a finite field, all constraint codes are identical, and our result reduces to the GLS result. Our formulation generalizes the GLS result to arbitrary group codes defined on graphs; *e.g.,* linear time-varying convolutional codes, linear tail-biting codes, or trellis codes over finite abelian groups.

We use a similar argument to provide a concise and transparent proof of the dual sum-product update rule stated in [3].

II. Codes, Realizations and Graphical Models

We follow the development and notation of [3].

Let $\{A_k, k \in \mathcal{I}_\mathcal{A}\}$ be a set of *symbol variables* A_k indexed by a discrete index set $\mathcal{I}_\mathcal{A}$, where each A_k is a finite abelian group. We will mostly consider symbol variables A_k that are vector spaces over a finite field $\mathbb{F}$, but all of our results and proofs generalize to arbitrary finite abelian groups.

A *group code* $\mathcal{C}$ is a subgroup of the Cartesian-product group $\mathcal{A} = \Pi_{k\in\mathcal{I}_\mathcal{A}} A_k$. If $\mathcal{A}$ is actually a vector space over a finite field $\mathbb{F}$, then a *linear code* $\mathcal{C}$ is a subspace of $\mathcal{A}$. From now on, all codes will be assumed to be group or linear codes.

A *generalized state realization* of a code $\mathcal{C} \subseteq \mathcal{A}$ is defined by a set of *state variables* $\{S_j, j \in \mathcal{I}_S\}$, and a set of *constraint codes* $\{\mathcal{C}_i, i \in \mathcal{I}_\mathcal{C}\}$, where $\mathcal{I}_S$ and $\mathcal{I}_\mathcal{C}$ are two further discrete index sets. Each state variable S_j is a finite group, or in the linear case a vector space over $\mathbb{F}$. Each constraint code $\mathcal{C}_i$ is a group or linear code involving certain subsets of the symbol and state variables. The *full behavior* of the realization is the set $\mathfrak{B} = (\mathbf{a}, \mathbf{s})$ of all configurations of symbol variables $\mathbf{a} \in \mathcal{A}$ and state variables $\mathbf{s} \in \mathcal{S} = \Pi_{j\in\mathcal{I}_S} S_j$ such that all constraints are satisfied. The *code* generated by the realization is the projection $\mathcal{C} = \mathfrak{B}_{|\mathcal{A}}$ of $\mathfrak{B}$ onto $\mathcal{A}$; *i.e.,* the set of all symbol configurations $\mathbf{a} \in \mathcal{A}$ that appear in some $(\mathbf{a}, \mathbf{s}) \in \mathfrak{B}$.

For example, in a *conventional state realization* of a linear code $\mathcal{C}$ over a finite field $\mathbb{F}$, 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 $\mathcal{I}_S$ 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 $k \in \mathcal{I}_S$. 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 Cartesian product 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 realization* is defined as a generalized state realization in which every symbol variable is involved in precisely one constraint code, and every state variable is involved in precisely two constraint codes. Thus a conventional state realization is normal. It is shown in [3] that any generalized state realization may be straightforwardly converted to a normal realization by introducing replication constraints, without essentially increasing the complexity of the realization.

A normal realization has a natural graphical model, in which each constraint code $\mathcal{C}_i$ corresponds to a vertex, each state variable S_j (which by definition is involved in two constraints) corresponds to an edge connecting the two corresponding constraint vertices, and each symbol variable 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 graph corresponding to a conventional state realization, which is a simple chain graph. Here vertices are represented by square boxes, and the "half-

978-1-4244-4983-5/09/$25.00 © 2009 IEEE

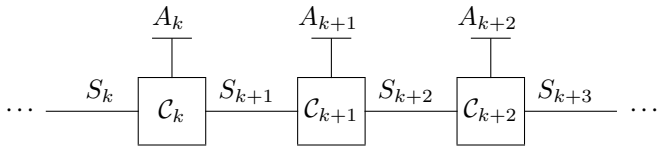

Fig. 1. Graph of a conventional state realization.

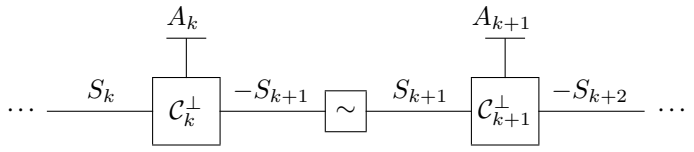

Fig. 2. Graph of dual of a conventional state realization, with sign inverter.

edges" corresponding to symbol variables are represented by special "dongle" symbols.

III. Dual Normal Realizations

The central duality result of [3] is the following: given a normal realization of a code $\mathcal{C}$, the dual normal realization generates the dual code $\mathcal{C}^\perp$. For simplicity of exposition, we will explain this result only for the case where $\mathcal{C}$ is a linear code over a finite field $\mathbb{F}$, but it holds also in the group case; see [3]. In the linear case, the dual code $\mathcal{C}^\perp$ is the usual orthogonal code to $\mathcal{C}$ under the usual symbolwise inner product.

We have seen that a normal realization for $\mathcal{C}$ is defined by a set of symbol variables $\{A_k, k \in \mathcal{I}_\mathcal{A}\}$, a set of state variables $\{S_j, j \in \mathcal{I}_S\}$, and a set of constraint codes $\{\mathcal{C}_i, i \in \mathcal{I}_\mathcal{C}\}$, where each symbol variable is involved in one constraint code, and each state variable is involved in two constraint codes.

The definition of a dual normal realization is slightly simpler in the case of a linear code $\mathcal{C}$ over the binary field $\mathbb{F}_2$ than in the general case, so we discuss the binary case first. Then the *dual normal realization* is defined by the same sets of symbol and state variables, and by the set of orthogonal constraint codes $\{\mathcal{C}_i^\perp, i \in \mathcal{I}_\mathcal{C}\}$, each involving the same variables as in the primal realization. The graph of the dual realization is thus the same as the graph of the primal realization, except that each constraint code $\mathcal{C}_i$ is replaced by its orthogonal code $\mathcal{C}_i^\perp$.

Example 1. 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 four-state realization as in Figure 1 in which each symbol variable A_k may be taken as $(\mathbb{F}_2)^2$, each state variable 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 generators

$$\begin{array}{c|c|l} 00 & 11 & 10; \\ 10 & 01 & 01; \\ 01 & 11 & 00, \end{array}$$

which represent the three nontrivial (state, symbol, next-state) transitions in the impulse response of the system. The orthogonal code $\mathcal{C}_k^\perp$ may easily be seen to be the $(6, 3)$ binary linear block code generated by the three generators

$$\begin{array}{c|c|l} 00 & 11 & 01; \\ 01 & 10 & 10; \\ 10 & 11 & 00, \end{array}$$

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 $\mathcal{C}^\perp$,[1] which in this case is again the code generated by $(1 + D + D^2, 1 + D^2)$. □

For a linear code $\mathcal{C}$ over a nonbinary field $\mathbb{F}$, one further trick (originally introduced by Mittelholzer [6] to dualize conventional state realizations over groups) is needed to define the dual normal realization: namely, in terms of the graph of the realization, insert a sign inverter in the middle of every edge. In other words, invert the sign of each state variable S_k in one of the two constraint codes in which it is involved. This is illustrated in Figure 2 for a conventional state realization.

Example 2 (*cf.* [1], [2]). 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 variable A_k may be taken as $(\mathbb{F}_3)^3$, each state variable 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

$$\begin{array}{c|c|l} 00 & 120 & 10; \\ 10 & 010 & 01; \\ 01 & 100 & 00; \\ 00 & 102 & 00, \end{array}$$

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

$$\begin{array}{c|c|l} 00 & 010 & 12; \\ 21 & 202 & 11; \\ 22 & 111 & 00, \end{array}$$

which represent the three nontrivial $(s'_k, a'_k, -s'_{k+1})$ transitions

[1] 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 $a(D)\tilde{b}(D) = 0$, where $\tilde{b}(D)$ is the D-transform of the time-reversed sequence $\tilde{\mathbf{b}} = \{b_{-k}, k \in \mathcal{I}_\mathcal{A}\}$.

in the impulse response of a conventional state realization of a single-input, three-output linear system over $\mathbb{F}_3$, with sign inverters as in Figure 2, 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.) 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 sequence-wise 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 $(1 + 2D, 1 + D^2, 1 + 2D)$.) □

IV. MacWilliams Identities

Given these duality results, various MacWilliams-type identities may be obtained in a more or less standard manner. We follow the development in [4].

Every finite abelian group $\mathcal{T}$ is a direct product of cyclic groups. In particular, every finite field $\mathbb{F}$ has $q = p^m$ elements for some prime p and is isomorphic as an additive group to $(\mathbb{Z}_p)^m$, and every vector space over a finite field $\mathbb{F}_{p^m}$ of dimension d is isomorphic to $(\mathbb{Z}_p)^{md}$. Thus, for some integer n, we may take $\mathcal{T} = (\mathbb{Z}_p)^n$, the set of n-tuples of elements of $\mathbb{Z}_p$.

Given a complex-valued function $\{x : \mathcal{T} \to \mathbb{C}, t \mapsto x(t)\}$ defined on $\mathcal{T} = (\mathbb{Z}_p)^n$, its (Fourier) *transform* is the complex-valued function $\{X : \mathcal{F} \to \mathbb{C}, f \mapsto X(f)\}$ defined on $\mathcal{F} = (\mathbb{Z}_p)^n$ by

$$X(f) = \sum_{\mathcal{T}} x(t)\omega^{f \cdot t}, \quad f \in \mathcal{F},$$

where ω is a primitive complex pth root of unity, and $f \cdot t \in \mathbb{Z}_p$ is the ordinary dot product between the n-tuples $f \in (\mathbb{Z}_p)^n$ and $t \in (\mathbb{Z}_p)^n$ over $\mathbb{Z}_p$. If we view $\mathbf{x} = \{x(t), t \in \mathcal{T}\}$ as a vector indexed by $\mathcal{T}$, and similarly $\mathbf{X} = \{X(f), f \in \mathcal{F}\}$ as a vector indexed by $\mathcal{F}$, then the transform can be expressed in matrix form as

$$\mathbf{X} = \mathcal{H}\mathbf{x},$$

where the *transform matrix* is $\mathcal{H} = \{\omega^{f \cdot t}, f \in \mathcal{F}, t \in \mathcal{T}\}$. Note that $\mathcal{H}^T = \mathcal{H}$, where $\mathcal{H}^T$ denotes the transpose of $\mathcal{H}$.

From the *orthogonality relation*

$$\sum_{\mathcal{F}} \omega^{f \cdot t} = \begin{cases} |\mathcal{F}|, & t = 0; \\ 0, & t \neq 0, \end{cases}$$

we obtain the matrix equation

$$\mathcal{H}\mathcal{H}^* = |\mathcal{F}| I_{|\mathcal{F}|},$$

where $\mathcal{H}^* = \{\omega^{-f \cdot t}, f \in \mathcal{F}, t \in \mathcal{T}\}$, $|\mathcal{F}| = |\mathcal{T}| = p^n$, and $I_{|\mathcal{F}|}$ is the $|\mathcal{F}| \times |\mathcal{F}|$ identity matrix. In other words, the inverse of $\mathcal{H}$ is $\mathcal{H}^{-1} = |\mathcal{F}|^{-1}\mathcal{H}^*$. Thus we obtain the *inverse transform*

$$\mathbf{x} = \mathcal{H}^{-1}\mathbf{X} = \frac{\mathcal{H}^*\mathbf{X}}{|\mathcal{F}|}.$$

We say that $\mathbf{x}$ and $\mathbf{X}$ are a *transform pair*.

We may extend these definitions to a set of indeterminates $\mathbf{z} = \{z(t), t \in \mathcal{T}\}$ indexed by $\mathcal{T}$, rather than a complex-valued function. The transform of this set is then a set of indeterminates $\mathbf{Z} = \{Z(f), f \in \mathcal{F}\}$ indexed by $\mathcal{F}$, where

$$\mathbf{Z} = \mathcal{H}\mathbf{z}.$$

Again, we have the inverse transform relationship

$$\mathbf{z} = \mathcal{H}^{-1}\mathbf{Z} = \frac{\mathcal{H}^*\mathbf{Z}}{|\mathcal{F}|},$$

and we say that $\mathbf{z}$ and $\mathbf{Z}$ are a transform pair.

For example, if $\mathcal{T} = \mathbb{Z}_2$, then $Z(0) = z(0) + z(1)$ and $Z(1) = z(0) - z(1)$; similarly, $z(0) = \frac{1}{2}(Z(0) + Z(1))$, and $z(1) = \frac{1}{2}(Z(0) - Z(1))$.

Now let us consider weight enumerators, initially for the case of a conventional state realization over a finite field $\mathbb{F}$ as in Figure 1. We define the *complete weight adjacency matrix* (CWAM) of each constraint code $\mathcal{C}_k \subseteq S_k \times A_k \times S_{k+1}$ as follows.

If $A_k = \mathbb{F}^n$, then define the complete weight enumerator of the n-tuple $\mathbf{a} = (a_1, \ldots, a_n \in \mathbb{F}^n)$ as the product $w(\mathbf{a}) = \Pi_{1 \leq i \leq n} w(a_i)$, where $\mathbf{w} = \{w(a), a \in \mathbb{F}\}$ is a set of indeterminates indexed by $\mathbb{F}$. Then the CWAM of $\mathcal{C}_k$ is the matrix $\Lambda(\mathbf{w}) = \{\Lambda(s_k, s_{k+1})(\mathbf{w}), (s_k, s_{k+1}) \in S_k \times S_{k+1}\}$ defined by

$$\Lambda(s_k, s_{k+1})(\mathbf{w}) = \sum_{\mathcal{C}(s_k, s_{k+1})} w(\mathbf{a}_k),$$

where $\mathcal{C}(s_k, s_{k+1}) = \{\mathbf{a}_k \mid (s_k, \mathbf{a}_k, s_{k+1}) \in \mathcal{C}_k\}$. Thus each entry $\Lambda(s_k, s_{k+1})(\mathbf{w})$ is a homogeneous integer polynomial of degree n in the $|\mathbb{F}|$ indeterminates $\mathbf{w} = \{w(a), a \in \mathbb{F}\}$.

Now let $\mathbf{y} = \{y(s_k), s_k \in S_k\}$ and $\mathbf{z} = \{z(s_{k+1}), s_{k+1} \in S_{k+1}\}$ be sets of indeterminates indexed by the state variables S_k and S_{k+1}, respectively, and define a *generating function* $g_{\Lambda(\mathbf{w})}(\mathbf{y}, \mathbf{z})$, a polynomial in the sets of indeterminates $\mathbf{y}$ and $\mathbf{z}$, as follows:

$$\begin{aligned} g_{\Lambda(\mathbf{w})}(\mathbf{y}, \mathbf{z}) &= \mathbf{y}^T \Lambda(\mathbf{w}) \mathbf{z} \\ &= \sum_{S_k \times S_{k+1}} y(s_k) \Lambda(s_k, s_{k+1})(\mathbf{w}) z(s_{k+1}). \end{aligned}$$

From the definition $\mathcal{C}(s_k, s_{k+1}) = \{\mathbf{a}_k \mid (s_k, \mathbf{a}_k, s_{k+1}) \in \mathcal{C}_k\}$, it follows that

$$g_{\Lambda(\mathbf{w})}(\mathbf{y}, \mathbf{z}) = \sum_{(s_k, \mathbf{a}_k, s_{k+1}) \in \mathcal{C}_k} y(s_k) w(\mathbf{a}_k) z(s_{k+1}).$$

Example 1 (cont.). The constraint code of Example 1 has the eight codewords $00|00|00, 00|11|10, 10|01|01, 10|10|11, 01|11|00, 01|00|10, 11|10|01, 11|01|11$, corresponding to the eight possible (state, symbol, next-state) transitions. Writing $\{w_0, w_1\}$ instead of $\{w(0), w(1)\}$, we see that we may write the CWAM of this constraint code in matrix form

as

$$\Lambda(\mathbf{w}) \quad = \quad \begin{array}{|c|c|c|c|c|} \hline s_k/s_{k+1} & 00 & 10 & 01 & 11 \\ \hline 00 & w_0^2 & w_1^2 & 0 & 0 \\ \hline 10 & 0 & 0 & w_0w_1 & w_0w_1 \\ \hline 01 & w_1^2 & w_0^2 & 0 & 0 \\ \hline 11 & 0 & 0 & w_0w_1 & w_0w_1 \\ \hline \end{array}$$

Equivalently, its generating function $g_{\Lambda(\mathbf{w})}(\mathbf{y}, \mathbf{z})$ is

$$\begin{aligned} g_{\Lambda(\mathbf{w})}(\mathbf{y}, \mathbf{z}) &= w_0^2(y_{00}z_{00} + y_{01}z_{10}) + w_1^2(y_{00}z_{10} + y_{10}z_{00}) \\ &+ w_0w_1(y_{01}z_{01} + y_{01}z_{11} + y_{11}z_{01} + y_{11}z_{11}). \end{aligned}$$

□

The key duality relation for MacWilliams identities is the *Poisson summation formula*, which says that "the sum of a function over a linear space is equal to the sum of the Fourier transform of the function over the dual space" [5]. For our case, this formula may be stated as follows:

Poisson summation formula. Let $\mathbf{x}$ and $\mathbf{X}$ be a transform pair defined on $\mathcal{T} = (\mathbb{Z}_p)^n$ and $\mathcal{F} = (\mathbb{Z}_p)^n$, respectively, and let $\mathcal{C}$ and $\mathcal{C}^\perp$ be orthogonal subgroups of $\mathcal{T}$ and $\mathcal{F}$, respectively. Then

$$\sum_{t \in \mathcal{C}} x(t) = \frac{1}{|\mathcal{C}^\perp|} \sum_{f \in \mathcal{C}^\perp} X(f).$$

Now, applying this formula to the equation above for $g_{\Lambda(\mathbf{w})}(\mathbf{y}, \mathbf{z})$, we obtain

$$\begin{aligned} g_{\Lambda(\mathbf{w})}(\mathbf{y}, \mathbf{z}) &= \sum_{\mathcal{C}_k} y(s_k)w(\mathbf{a}_k)z(s_{k+1}) \\ &= \frac{1}{|\mathcal{C}_k^\perp|} \sum_{\mathcal{C}_k^\perp} Y(\hat{s}_k)W(\hat{\mathbf{a}}_k)Z(-\hat{s}_{k+1}). \end{aligned}$$

Here we use the fact that the transform of a product is the product of their transforms, where $\mathbf{Y} = \mathcal{H}_y\mathbf{y}$, $\mathbf{W} = \mathcal{H}_w\mathbf{w}$, and $\mathbf{Z} = \mathcal{H}_z\mathbf{z}$. Note that $\mathbf{W}$ is itself a product transform. Also, since the elements of $\mathcal{H}_z$ are $\mathcal{H}_z(s_{k+1}, -\hat{s}_{k+1}) = \omega^{-s_{k+1} \cdot \hat{s}_{k+1}}$, the matrix $\mathcal{H}_z$ is the conjugate of the usual transform matrix over S_{k+1}.

If we define the CWAM $\hat{\Lambda}(\mathbf{W})$ of $\mathcal{C}_k^\perp$ and its generating function $g_{\hat{\Lambda}(\mathbf{W})}(\mathbf{Y}, \mathbf{Z})$ similarly to the analogous quantities for $\mathcal{C}_k$, then we obtain

$$g_{\hat{\Lambda}(\mathbf{W})}(\mathbf{Y}, \mathbf{Z}) = \sum_{\mathcal{C}_k^\perp} Y(\hat{s}_k)W(\hat{\mathbf{a}}_k)Z(-\hat{s}_{k+1}).$$

Using inverse transforms, we thus obtain

$$\begin{aligned} g_{\hat{\Lambda}(\mathbf{W})}(\mathbf{Y}, \mathbf{Z}) &= |\mathcal{C}_k^\perp| g_{\Lambda(\mathbf{w})}(\mathbf{y}, \mathbf{z}) \\ &= |\mathcal{C}_k^\perp| g_{\Lambda(\mathcal{H}_w^{-1}\mathbf{W})}(\mathcal{H}_y^{-1}\mathbf{Y}, \mathcal{H}_z^{-1}\mathbf{Z}). \end{aligned}$$

This MacWilliams identity shows how the generating function for the CWAM $\hat{\Lambda}(\mathbf{W})$ of $\mathcal{C}_k^\perp$ may be obtained from that for $\mathcal{C}_k$, or *vice versa*.

Alternatively, since

$$g_{\hat{\Lambda}(\mathbf{W})}(\mathbf{Y}, \mathbf{Z}) = \mathbf{Y}^T \hat{\Lambda}(\mathbf{W}) \mathbf{Z}$$

and

$$g_{\Lambda(\mathbf{w})}(\mathbf{y}, \mathbf{z}) = \mathbf{y}^T \Lambda(\mathbf{w}) \mathbf{z} = \mathbf{Y}^T \mathcal{H}_y^{-1} \Lambda(\mathcal{H}_w^{-1}\mathbf{W}) \mathcal{H}_z^{-1} \mathbf{Z},$$

we may simply write

$$\hat{\Lambda}(\mathbf{W}) = |\mathcal{C}_k^\perp| \mathcal{H}_y^{-1} \Lambda(\mathcal{H}_w^{-1}\mathbf{W}) \mathcal{H}_z^{-1},$$

a MacWilliams identity that shows how the CWAM of $\mathcal{C}_k^\perp$ may be obtained from the CWAM of $\mathcal{C}_k$.

Example 1 (cont.). Given the CWAM $\Lambda(\mathbf{w})$ of the constraint code $\mathcal{C}_k$ of Example 1, the CWAM $\hat{\Lambda}(\mathbf{W})$ of the orthogonal constraint code $\mathcal{C}_k^\perp$ is given by the matrix equation at the top of the next page, where we have substituted the dual indeterminates W_0 and W_1 for $w_0 + w_1$ and $w_0 - w_1$. □

The Hamming weight adjacency matrix (HWAM) Λ_H of a constraint code $\mathcal{C}_k$ is obtained by substituting 1 for $w(0)$ and w for each $w(a), a \neq 0$. Thus each element $\Lambda_H(s_k, s_{k+1})(w)$ becomes a polynomial of degree n in the single indeterminate w. The dual indeterminates become $W(0) = 1 + (|\mathbb{F}| - 1)w$ and $W(a) = 1 - w, a \neq 0$, which scale to 1 and $W = (1 - w)/(1 + (|\mathbb{F}| - 1)w)$, respectively. Substituting in the above MacWilliams-type identities for CWAMs, we obtain MacWilliams-type identities for HWAMs. This yields the main result of [1], [2].[2]

Example 2 (cont.). For a worked-out example of the HWAM $\hat{\Lambda}(W)$ of the orthogonal code $\mathcal{C}_k^\perp$ to the constraint code of Example 2, see [2]. □

Although our development has focussed on conventional state realizations of linear time-invariant convolutional codes, it may be straightforwardly extended to obtain MacWilliams identities for any generalized state realization of any finite abelian group code defined on an arbitrary graph, because constraint code duality holds in the general case.

V. Dualizing the Sum-Product Update Rule

Another duality result in [3] is a general method for dualizing the sum-product update rule, which among other things yields the "tanh rule" of APP decoding. The approach of this paper yields a cleaner derivation of this result.

Again, for simplicity we restrict attention to conventional state realizations, in which each constraint code $\mathcal{C}_k$ specifies the state transitions in $S_k \times A_k \times S_{k+1}$ that can possibly occur. Let the (right-going) *message* be any real- or complex-valued function $\mathbf{m}_k = \{m_k(s_k), s_k \in S_k\}$ of the state variable S_k, and let $\mathbf{f}_k = \{f_k(a_k), a_k \in A_k\}$ be any real-or complex-valued *weight function* of the symbol variable A_k. Then the *sum-product update rule* associated with constraint code $\mathcal{C}_k$ is

$$m_{k+1}(s_{k+1}) = \sum_{\mathcal{C}_k(s_{k+1})} m_k(s_k) f_k(a_k),$$

[2]The MacWilliams identity of [1], [2] 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.

$$\frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} w_0^2 & w_1^2 & 0 & 0 \\ 0 & 0 & w_0w_1 & w_0w_1 \\ w_1^2 & w_0^2 & 0 & 0 \\ 0 & 0 & w_0w_1 & w_0w_1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} = \begin{bmatrix} W_0^2 & 0 & W_1^2 & 0 \\ W_1^2 & 0 & W_0^2 & 0 \\ 0 & W_0W_1 & 0 & W_0W_1 \\ 0 & W_0W_1 & 0 & W_0W_1 \end{bmatrix}$$

where $\mathcal{C}_k(s_{k+1}) = \{(s_k, a_k) \in S_k \times A_k \mid (s_k, a_k, s_{k+1}) \in \mathcal{C}_k\}$. In other words, if we define a set of indeterminates $\mathbf{x} = \{x(s_{k+1}), s_{k+1} \in S_{k+1}\}$, then $m_{k+1}(s_{k+1})$ is the coefficient of $x(s_{k+1})$ in the homogeneous degree-1 multivariate generating function $g_{k+1}(\mathbf{x})$ defined by

$$g_{k+1}(\mathbf{x}) = \mathbf{m}_{k+1}^T\mathbf{x} = \sum_{s_{k+1}} m_{k+1}(s_{k+1})x(s_{k+1}).$$

From the definition of $\mathcal{C}_k(s_{k+1})$, it follows that

$$g_{k+1}(\mathbf{x}) = \sum_{\mathcal{C}_k} m_k(s_k)f_k(a_k)x(s_{k+1}).$$

Using the Poisson summation formula, we obtain

$$\begin{aligned} g_{k+1}(\mathbf{x}) &= \sum_{\mathcal{C}_k} m_k(s_k)f_k(a_k)x(s_{k+1}) \\ &= \frac{1}{|\mathcal{C}_k^\perp|}\sum_{\mathcal{C}_k^\perp} M_k(\hat{s}_k)F_k(\hat{a}_k)X(-\hat{s}_{k+1}) \\ &= \frac{\hat{g}_{k+1}(\mathbf{X})}{|\mathcal{C}_k^\perp|}, \end{aligned}$$

where we again use the fact that the transform of a product is the product of their transforms, and define transformed functions or indeterminates by corresponding capitalized functions or indeterminates. The left side of this equation is the generating function $g_{k+1}(\mathbf{x})$ of the message $\{m_{k+1}\}$, and the right side is (up to scale) the generating function $\hat{g}_{k+1}(\mathbf{X})$ of the message $\mathbf{M}_{k+1}$ obtained by performing the sum-product update algorithm for $\mathcal{C}_k^\perp$ upon the message $\mathbf{M}_k$ and the weight function $\mathbf{F}_k$. Moreover, the messages $\mathbf{m}_{k+1}$ and $\mathbf{M}_{k+1}$ form a transform pair.

Consequently, we have the following recipe for performing the sum-product update rule for $\mathcal{C}_k$:

1) Transform the incoming messages $\mathbf{m}_k$ and $\mathbf{f}_k$ to $\mathbf{M}_k$ and $\mathbf{F}_k$;
2) Perform the sum-product update rule for $\mathcal{C}_k^\perp$ to generate an output message $\mathbf{M}_{k+1}$;
3) Inverse transform $\mathbf{M}_{k+1}$ to obtain the message $\mathbf{m}_{k+1}$, up to the scale factor $|\mathcal{C}_k^\perp|$.

Since the complexity of performing the sum-product update rule for $\mathcal{C}_k$ is proportional to $|\mathcal{C}_k|$, this dual computation may be attractive if $|\mathcal{C}_k^\perp| < |\mathcal{C}_k|$.

Example 3 ("tanh rule"). Let S_k, A_k and S_{k+1} be binary variables taking values in $\mathbb{F}_2$, and let $\mathcal{C}_k$ be the $(3, 2)$ single-parity-check code consisting of the four codewords $(000, 011, 101, 110)$; then $\mathcal{C}_k^\perp$ is the $(3, 1)$ repetition code consisting of the two codewords $(000, 111)$. Let the incoming message and weight function be $\mathbf{m}_k = (m_0, m_1)$ and $\mathbf{f}_k = (f_0, f_1)$; then the transformed message and weight function are $\mathbf{M}_k = (M_0 = m_0 + m_1, M_1 = m_0 - m_1)$ and $\mathbf{F}_k = (F_0 = f_0 + f_1, F_1 = f_0 - f_1)$. Using two multiplications, the sum-product update equation then produces the message $\mathbf{M}_{k+1} = (M_{k+1}(0) = (m_0 + m_1)(f_0 + f_1), M_{k+1}(1) = (m_0 - m_1)(f_0 - f_1))$. Thus, up to scale, the message $\mathbf{m}_{k+1}$ is

$$\begin{aligned} m_{k+1}(0) &= M_{k+1}(0) + M_{k+1}(1) \propto m_0f_0 + m_1f_1; \\ m_{k+1}(1) &= M_{k+1}(0) - M_{k+1}(1) \propto m_0f_1 + m_1f_0; \end{aligned}$$

which is evidently the message that would have been computed by a direct computation of the sum-product update rule for $\mathcal{C}_k$, which requires four multiplications. □

Again, although our development has focussed on a constraint code of a conventional linear state realization, it may be straightforwardly extended to obtain a dual sum-product update rule for an arbitrary constraint code over any finite abelian group.

ACKNOWLEDGMENT

For comments on an earlier version of this paper, I am grateful to H. Gluesing-Luerssen.

REFERENCES

[1] H. Gluesing-Luerssen and G. Schneider, "On the MacWilliams identity for convolutional codes," *IEEE Trans. Inform. Theory*, vol. 54, pp. 1536–1550, April 2008. ArXiv: cs/0603013.

[2] 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. ArXiv: 0805.3484v1 [cs.IT].

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

[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] J. H. Conway and N. J. A. Sloane, *Sphere Packings, Lattices and Groups*. New York: Springer-Verlag, 1988.

[6] T. Mittelholzer, "Convolutional codes over groups: A pragmatic approach," in *Proc. 33d Allerton Conf. Commun. Contr. Comput.* (Allerton, IL), pp. 380–381, Sept. 1995.