Miscelánea Matemática **42** (2006) 27–50 SMM

Una nueva manera de explicar la decodificación, para los códigos de Reed-Muller, que sólo requiere conocimientos de teoría de conjuntos

Gerardo Vega

Dirección General de Servicios de Cómputo Académico

Universidad Nacional Autónoma de México

04510 México D.F.

México

gerardov@servidor.unam.mx

Resumen

El presente trabajo pretende ser un auxiliar didáctico para un curso introductorio a las funciones booleanas, a los códigos de Reed-Muller y los algoritmos de codificación y decodificación que le son aplicables en el caso binario. Este material está dividido en cuatro partes; en la primera parte se describen las funciones booleanas así como algunos de sus resultados más relevantes. En la segunda parte, se emplean estos resultados para presentar, de manera natural, a los códigos de Reed-Muller. La tercera parte se dedica a la descripción de los algoritmos de codificación y decodificación que son aplicables a tales códigos. Es de notar que, en general, estos algoritmos se describen en términos de geometría algebraica, sin embargo, para este trabajo se emplea sólo teoría de conjuntos. Lo anterior además de facilitar su entendimiento, constituye un componente totalmente original en este material. Por último, en el apéndice de este trabajo se presenta el código fuente, en lenguaje "C", de una implementación para los algoritmos de codificación y decodificación asociados al código de Reed-Muller de orden 2 en 5 variables ($\mathcal{R}(2,5)$).

Palabras Clave: Funciones Booleanas, Códigos de Reed-Muller, Reed-algorithm.

1. Funciones Booleanas

El conjunto de funciones booleanas son de importancia debido a que, entre otras cosas, es a través de este tipo de funciones que es posible definir una familia de códigos lineales muy importantes conocidos como *códigos de Reed-Muller*. Este tipo de códigos fueron de los primeros en ser estudiados ampliamente y también fueron de los primeros que se emplearon en diversas aplicaciones prácticas, por ejemplo, la sonda espacial *Mariner 9* utilizó códigos de Reed-Muller para codificar las imágenes que se enviaban a la Tierra.

Para introducir este tipo de funciones, recordemos que para el caso del campo de los números reales, $\mathbb{R}$, y para cualquier entero m mayor que cero, es común trabajar con funciones del tipo: $f : \mathbb{R}^m \to \mathbb{R}$. Por ejemplo, para $m = 3$, f podría estar dada por:

$$f(v_1, v_2, v_3) = v_1 + 3 + v_1v_2 - 9.7(v_1^2v_2v_3) \ ,$$

donde $(v_1, v_2, v_3) \in \mathbb{R}^3$. Análogamente, si nuestro campo es $GF(2)$, entonces tiene sentido considerar funciones del tipo $f : GF(2)^m \to GF(2)$. Por ejemplo, para $m = 3$, f podría estar dada por:

$$f(v_1, v_2, v_3) = v_1 + 1 + v_1v_2 + v_1v_2v_3 \ ,$$

donde $(v_1, v_2, v_3) \in GF(2)^3$. Obsérvese que, debido a las propiedades de $GF(2)$, los escalares en la función anterior sólo podrán ser cero o uno. Por la misma razón, es claro que $x^2 = x$, para toda $x \in GF(2)$, y por tanto $v_1^2v_2v_3 = v_1v_2v_3$. Por último, se debe mantener presente que es indistinto sumar o restar algebraicamente este tipo de funciones.

En forma general y para una $m \in \mathbb{N}$ llamaremos *función booleana* a toda función de la forma $f : GF(2)^m \to GF(2)$ y denotaremos por $\mathcal{B}_m$ al *conjunto de funciones booleanas de m argumentos*, es decir:

$$\mathcal{B}_m = \{f | f : GF(2)^m \to GF(2)\} \ .$$

Debe observarse que $|\mathcal{B}_m| < \infty$, además $\mathcal{B}_m$ es un espacio vectorial sobre $GF(2)$ de dimensión finita en donde una base para este espacio es:

$$B = \{1, v_1, v_2, \ldots, v_m, v_1v_2, \ldots, v_1v_m, v_2v_3, \ldots, v_2v_m, \ldots, v_1v_2 \cdots v_m\} \ . \tag{1}$$

Ejemplo 1.1. Para $m = 3$ se tendría que:

$$B = \{1, v_1, v_2, v_3, v_1v_2, v_1v_3, v_2v_3, v_1v_2v_3\} .$$

Note cómo es posible hacer una asociación uno a uno entre los elementos en la base B de $\mathcal{B}_m$ y los elementos del conjunto potencia de $\{v_1, v_2, \ldots, v_m\}$, por lo que entonces en general se tiene que:

$$|B| = \left(\begin{array}{c} m \\ 0 \end{array} \right) + \left(\begin{array}{c} m \\ 1 \end{array} \right) + \cdots + \left(\begin{array}{c} m \\ m \end{array} \right) = 2^m ,$$

y por tanto:

$$|\mathcal{B}_m| = 2^{2^m} .$$

Por ejemplo:

$$|\mathcal{B}_m| = \begin{cases} 4 & \text{si } m = 1 \\ 16 & \text{si } m = 2 \\ 256 & \text{si } m = 3 \\ 65536 & \text{si } m = 4 \\ 2^{32} & \text{si } m = 5 \end{cases} .$$

A través de lo que se conoce como descomposición o representación binaria de un número, es posible establecer una biyección entre el conjunto de vectores en $GF(2)^m$ y conjunto de enteros en $\{0, 1, 2, \ldots, 2^m - 1\}$. Más específicamente, dado el vector $\vec{b} = (b_1, b_2, \ldots, b_m)$ en $GF(2)^m$, le asociaremos a éste el entero b en $\{0, 1, 2, \ldots, 2^m - 1\}$, que se obtiene como $b = b_1 2^{m-1} + b_2 2^{m-2} + \cdots + b_m$, es decir:

$$\vec{b} = (b_1, b_2, \ldots, b_m) \longleftrightarrow b = b_1 2^{m-1} + b_2 2^{m-2} + \cdots + b_m ,$$

donde la suma y producto a la derecha de la relación anterior se toman sobre $\mathbb{Z}$. En el marco de esta biyección, al vector $\vec{b}$ se le llamará *la Representación Binaria* del entero b. Por ejemplo si $m = 3$, entonces:

$$\begin{array}{cccc} \vec{b} = (b_1, & b_2, & b_3) & \longleftrightarrow b \\ (0, & 0, & 0\ ) & \longleftrightarrow 0 \\ (0, & 0, & 1\ ) & \longleftrightarrow 1 \\ (0, & 1, & 0\ ) & \longleftrightarrow 2 \\ (0, & 1, & 1\ ) & \longleftrightarrow 3 \end{array} \qquad \begin{array}{cccc} \vec{b} = (b_1, & b_2, & b_3) & \longleftrightarrow b \\ (1, & 0, & 0\ ) & \longleftrightarrow 4 \\ (1, & 0, & 1\ ) & \longleftrightarrow 5 \\ (1, & 1, & 0\ ) & \longleftrightarrow 6 \\ (1, & 1, & 1\ ) & \longleftrightarrow 7 \end{array} .$$

Notación 1.2. Sea $x \in \{0, 1, 2, \ldots, 2^m - 1\}$ y sea $f \in \mathcal{B}_m$, entonces se entenderá por $f(x)$, al elemento en $GF(2)$ que resulta de la evaluación de la función booleana f en el vector binario de longitud m que corresponda a la representación binaria de x.

Ejemplo 1.3. Si $m = 3$ y $f = v_1 + 1 + v_2v_3 + v_1v_2v_3$, entonces $f(3) := f(0, 1, 1) = 0 + 1 + 1 \cdot 1 + 0 \cdot 1 \cdot 1 = 1 + 1 = 0$.

Definición 1.4. Sea $f \in \mathcal{B}_m$, entonces definimos a la tabla de verdad de f, $T_v(f)$, como el vector binario de longitud 2^m dado por:

$$T_v(f) = (f(0), f(1), f(2), \ldots, f(2^m - 1)) \, .$$

Ejemplo 1.5. Para la función booleana del ejemplo anterior;

$$T_v(f) = (1, 1, 1, 0, 0, 0, 0, 0) \, .$$

Ya se dijo que $\mathcal{B}_m$ tiene estructura de espacio vectorial, sin embargo, podemos enriquecer aún más esta estructura si definimos un producto para él. La manera natural para definir este producto es a través del producto algebraico. Por ejemplo, si $f_1, f_2 \in \mathcal{B}_3$, dadas por $f_1 = 1 + v_2 + v_1v_2$ y $f_2 = v_1v_3 + v_2$, entonces el producto f de f_1 y f_2, estaría dado por:

$$f = (1 + v_2 + v_1v_2)(v_1v_3 + v_2) = v_1v_3 + v_1v_2v_3 + v_1^2v_2v_3 + v_2 + v_2^2 + v_1v_2^2 \, ,$$

sin embargo, recordemos que $x^2 = x$ para todo $x \in GF(2)$ y por tanto $v_1v_2v_3 + v_1^2v_2v_3 = 0$, $v_2 + v_2^2 = 0$ y $v_1v_2^2 = v_1v_2$. De esta manera, se tiene que f está dada simplemente por:

$$f = v_1v_3 + v_1v_2 \, .$$

Observe cómo, sin importar la expresión algebraica inicial de una función booleana, siempre será posible re-expresar a tal función como una combinación lineal, sobre $GF(2)$, de elementos en la base B, dada por (1). Para cada función booleana, esta expresión final en términos de la base B, es única (debido precisamente a que B es una base) y se le conoce como la *forma normal algebraica* de una función booleana.

Claramente el producto algebraico anterior es cerrado en $\mathcal{B}_m$. Consecuentemente, esto nos permite ahora afirmar que $\mathcal{B}_m$ ha adquirido estructura de anillo conmutativo con elemento unitario, donde la unidad estaría dada por la función booleana constante $f = 1$. Ahora bien, observe que $(1 + v_1)v_1v_2 = v_1v_2 + v_1v_2 = 0$. Por tanto, podemos concluir, de una manera aún más precisa, que $\mathcal{B}_m$ es un anillo conmutativo con elemento unitario y con divisores de cero.

Ejercicio 1.6. Ya se vió con anterioridad que $(1+v_2+v_1v_2)(v_1v_3+v_2) = v_1v_3 + v_1v_2$ en $\mathcal{B}_3$, compruebe ahora directamente que:

$$T_v((1 + v_2 + v_1v_2)(v_1v_3 + v_2)) = T_v(v_1v_3 + v_1v_2) \ .$$

Se ha visto ya que la tabla de verdad de una función booleana en $\mathcal{B}_m$ es un vector binario en $GF(2)^{2^m}$. La pregunta que surge es: ¿Dado un vector binario $\vec{y}$ en $GF(2)^{2^m}$, existirá una función booleana f en $\mathcal{B}_m$ tal que $T_v(f) = \vec{y}$? La respuesta a esta pregunta es afirmativa y la manera de encontrar tal función booleana es a través de una construcción muy sencilla, la cual ilustraremos con el siguiente ejemplo:

Ejemplo 1.7. Para $m = 3$ y $\vec{y} = (0, 0, 0, 1, 0, 0, 1, 1) \in GF(2)^8$, queremos encontrar $f \in \mathcal{B}_3$ tal que $T_v(f) = \vec{y}$. Lo que se quiere, en otras palabras, es encontrar f tal que $f(3) = f(6) = f(7) = 1$ y que $f(0) = f(1) = f(2) = f(4) = f(5) = 0$. Ahora bien, observe que la representación binaria de los números 3, 6 y 7 son $(0, 1, 1)$, $(1, 1, 0)$ y $(1, 1, 1)$, respectivamente. A continuación asociamos a la variable v_1 con la primera entrada de estas representaciones, a v_2 con la segunda entrada y a v_3 con la tercera. Teniendo en mente estas asociaciones, podemos ahora, a su vez, asociar a la primera representación binaria, $(0, 1, 1)$, el término $\bar{v}_1v_2v_3$, donde estamos definiendo $\bar{v} := 1 + v$ para todo $v \in GF(2)$. De manera análoga, los términos $v_1v_2\bar{v}_3$ y $v_1v_2v_3$ se asociarián con la segunda y tercera representaciones binarias (i.e. $(1, 1, 0)$ y $(1, 1, 1)$) respectivamente. Usando estos términos podemos construir la siguiente función booleana:

$$f = \bar{v}_1v_2v_3 + v_1v_2\bar{v}_3 + v_1v_2v_3 = (1 + v_1)v_2v_3 + v_1v_2(1 + v_3) + v_1v_2v_3 \ .$$

Esta función booleana, así construida, permite verificar fácilmente que $T_v(f) = \vec{y}$, más aún, si desarrollamos la simplificación de f, podemos comprobar que ésta puede ser re-expresada en su forma normal algebraica simplemente como:

$$f = v_1v_2 + v_2v_3 + v_1v_2v_3 \ .$$

Ejercicio 1.8. Compruebe que las dos expresiones algebraicas para f, en el ejemplo anterior, tienen efectivamente la misma tabla de verdad y que ésta es igual al vector $\vec{y}$.

El número de vectores binarios de longitud 2^m es 2^{2^m} (i.e. $|GF(2)^{2^m}| = 2^{2^m}$), el cual es exactamente igual al número de funciones booleanas de m variables, $\mathcal{B}_m$. Ahora bien, por la manera en que se ha definido

a la función T_v, debe ser claro que a cada función booleana le corresponde una, y sólo una, tabla de verdad. Por otro lado, la unicidad en la forma normal algebraica de cada función booleana, determina, recíprocamente, que a cada tabla de verdad le corresponde una, y sólo una, función booleana en su forma normal algebraica. Esto último nos permite concluir que la función:

$$T_v : \mathcal{B}_m \longrightarrow GF(2)^{2^m} \ ,$$

es biyectiva, es decir existe una relación uno a uno entre las funciones booleanas y las tablas de verdad. Por esta razón, de aquí en adelante, cada vez que se tenga un vector binario en $GF(2)^{2^m}$, se verá éste como la función booleana en m argumentos, y recíprocamente, cada vez que se tenga una función booleana en m argumentos, se verá ésta como un vector binario en $GF(2)^{2^m}$.

Ahora bien, si $f_1, f_2 \in \mathcal{B}_m$ entonces, ¿cuánto será $T_v(f_1 + f_2)$? Veamos pues:

$$\begin{aligned} T_v(f_1 + f_2) &= ((f_1 + f_2)(0), (f_1 + f_2)(1), \ldots, (f_1 + f_2)(2^m - 1)) \\ &= (f_1(0) + f_2(0), f_1(1) + f_2(1), \ldots, f_1(2^m - 1) + f_2(2^m - 1)) \\ &= (f_1(0), f_1(1), \ldots, f_1(2^m - 1)) + (f_2(0), f_2(1), \ldots, f_2(2^m - 1)) \ , \end{aligned}$$

es decir, tenemos que $T_v(f_1 + f_2) = T_v(f_1) + T_v(f_2)$. Haciendo el caso análogo para el producto de funciones booleanas, se obtiene que:

$$T_v(f_1 f_2) = (f_1(0)f_2(0), f_1(1)f_2(1), \ldots, f_1(2^m - 1)f_2(2^m - 1)) \ .$$

Observe cómo la expresión del lado derecho de la ecuación anterior, corresponde al producto punto entre $T_v(f_1)$ y $T_v(f_2)$, pero sin realizar las sumas.

Definición 1.9. Sean $\vec{x}=(x_1, x_2, \ldots, x_n)$, $\vec{y}=(y_1, y_2, \ldots, y_n) \in GF(2)^n$, entonces definiremos el producto "$*$"en $GF(2)^n$ como:

$$\vec{x} * \vec{y} = (x_1 y_1, x_2 y_2, \ldots, x_n y_n) \ .$$

donde los productos en el lado derecho de la ecuación anterior, se toman sobre $GF(2)$.

Con la suma usual sobre $GF(2)^{2^m}$ y ahora con la introducción del producto "$*$"en $GF(2)^n$, y particularmente sobre $GF(2)^{2^m}$, se tiene entonces que $GF(2)^{2^m}$ ha adquirido estructura de anillo conmutativo. Por

otro lado, ya se dijo que $\mathcal{B}_m$ es también un anillo y como la biyección; $T_v : \mathcal{B}_m \to GF(2)^{2^m}$, es tal que:

$$\begin{aligned} T_v(f_1 + f_2) &= T_v(f_1) + T_v(f_2) \\ T_v(f_1 f_2) &= T_v(f_1) * T_v(f_2) \ , \end{aligned}$$

para toda $f_1, f_2 \in \mathcal{B}_m$, entonces podemos concluir que la biyección T_v es realmente un *isomorfismo de anillos* entre $\mathcal{B}_m$ y $GF(2)^{2^m}$. Lo anterior, reafirma la idea que se tenía sobre la factibilidad de operar indistintamente con funciones booleanas o con sus tablas de verdad, pues ahora se ve claramente que lo que se haga en un anillo se reflejará "idénticamente"en el otro anillo.

Definición 1.10. Sea $m \in \mathbb{N} \setminus \{0\}$ y sea $S = \{1, 2, \ldots, m\}$. Para cada $\alpha \in 2^S$ (2^S es el conjunto potencia de S y por tanto $\alpha \subseteq S$), definimos la función booleana, μ^α, la cual llamaremos monomio en α, como:

$$\mu^\alpha := \begin{cases} \prod_{i \in \alpha} v_i & \text{si } \alpha \neq \phi \\ 1 & \text{de otro modo} \end{cases} .$$

Ejemplo 1.11. Si $m = 5$ y $\alpha = \{2, 4, 5\}$, entonces:

$$\mu^\alpha = \mu^{\{2,4,5\}} = \prod_{i \in \{2,4,5\}} v_i = v_2 v_4 v_5 \ .$$

La notación anterior permite una expresión más compacta para (1):

$$B = \{\mu^\alpha | \alpha \in 2^S\} \ ,$$

de donde se sigue claramente que $|B| = |2^S| = 2^{|S|} = 2^m$. También esta notación permite dar una definición más precisa de $\mathcal{B}_m$, a saber:

$$\mathcal{B}_m = \{f | f = \sum_{\alpha \in I} \mu^\alpha \ ; \ I \subseteq 2^S\} \ .$$

Ejemplo 1.12. Si $m = 5$ y $I = \{\phi, \{1, 2, 5\}, \{3, 4\}, \{1\}, \{1, 2, 3, 5\}\}$, entonces:

$$f = \mu^\phi + \mu^{\{1,2,5\}} + \mu^{\{3,4\}} + \mu^{\{1\}} + \mu^{\{1,2,3,5\}} = 1 + v_1 v_2 v_5 + v_3 v_4 + v_1 + v_1 v_2 v_3 v_5 \ .$$

Nuevamente, debe ser claro como $|\mathcal{B}_m| = |2^{2^S}| = 2^{|2^s|} = 2^{2^m}$. Un concepto importante de toda función booleana es su grado, el cual podemos definir fácilmente en términos de la notación de monomios.

Definición 1.13. Sea $I \subseteq 2^S$ y sea f la función booleana dada por $f = \sum_{\alpha \in I} \mu^\alpha$, entonces definimos el grado de f, denotado por grado(f), como:

$$\text{grado}(f) := \text{máx}\{|\alpha| \mid \alpha \in I\} .$$

Ejemplo 1.14. Para la función booleana del ejemplo anterior, se tiene que grado $(f) = 4$.

Notación 1.15. Sea $p \in \{0, 1, 2, \ldots, 2^m - 1\}$ y sea $\alpha \in 2^S$, entonces se entenderá por $(\mu^\alpha)_p$, como la evaluación de la función booleana μ^α en p, es decir:

$$(\mu^\alpha)_p := \mu^\alpha(p) .$$

Ejemplo 1.16. Si $m = 5$, $\alpha = \{1, 2, 5\}$ y $p = 19$, entonces dado que la representación binaria de 19 es $(1, 0, 0, 1, 1)$, se tiene que:

$$(\mu^\alpha)_p = (\mu^{\{1,2,5\}})_{19} = v_1 v_2 v_5(19) = 1 \cdot 0 \cdot 1 = 0 .$$

Definición 1.17. Con la notación anterior, sea $\alpha \in 2^S$. Definimos al nulificador de α, N_α, como el siguiente conjunto de vectores binarios en $GF(2)^m$:

$$N_\alpha = \{\vec{p} = (p_1, p_2, \ldots, p_m) \in GF(2)^m \mid p_i = 0 \quad \forall \ i \in \alpha\} . \qquad (2)$$

Ejemplo 1.18. Si $m = 5$ y $\alpha = \{1, 2, 5\}$, entonces N_α estará conformado por los siguientes cuatro vectores binarios:

$$N_\alpha = \begin{array}{c} v_1 \quad v_2 \quad v_3 \quad v_4 \quad v_5 \\ \left\{ \begin{array}{lllll} (0, & 0, & 0, & 0, & 0), \\ (0, & 0, & 0, & 1, & 0), \\ (0, & 0, & 1, & 0, & 0), \\ (0, & 0, & 1, & 1, & 0) \end{array} \right\} \end{array} . \qquad (3)$$

Observe cómo, de manera general, se cumple que $|N_\alpha| = 2^{m-|\alpha|}$. Ahora bien, si aprovechamos la relación entre los números enteros en $\{0, 1, 2, \ldots, 2^m - 1\}$ y su respectivas descomposiciones binarias, es posible lograr, si nos permitimos un pequeño abuso de notación, una descripción más compacta de los elementos en N_α. Es decir, en lugar de listar todos los vectores binarios $\vec{p}$ en $GF(2)^m$ que satisfacen (2), escribiremos simplemente el conjunto de enteros cuyas descomposiciones binarias corresponden a los vectores en esta lista.

Ejemplo 1.19. Para el caso del ejemplo anterior, tendríamos que N_α estaría dado por:

$$N_\alpha = \{0, 2, 4, 6\} .$$

Veamos ahora algunas propiedades de los monomios y de N_α:

Propiedad 1.20. Sean m, S y α como antes, y sean p, b y c enteros tales que $c = p + b$ y donde $\vec{p} \in N_\alpha$ y $\vec{b} \in N_{\bar{\alpha}}$, con $\bar{\alpha} = S \setminus \alpha$. Entonces $\vec{p} + \vec{b} = \vec{c}$.

Demostración: Es suficiente una prueba a través del siguiente ejemplo. □

Ejemplo 1.21. Para $m = 5$ con $\alpha = \{1, 2, 5\}$, entonces $\bar{\alpha} = \{3, 4\}$ y por tanto:

$$N_{\bar{\alpha}} = \left\{ \begin{array}{ccccc|ccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_1 & v_2 & v_3 & v_4 & v_5 \\ (0, & 0, & 0, & 0, & 0), & (1, & 0, & 0, & 0, & 0), \\ (0, & 0, & 0, & 0, & 1), & (1, & 0, & 0, & 0, & 1), \\ (0, & 1, & 0, & 0, & 0), & (1, & 1, & 0, & 0, & 0), \\ (0, & 1, & 0, & 0, & 1), & (1, & 1, & 0, & 0, & 1) \end{array} \right\} . \quad (4)$$

De una inspección directa de las ecuaciones (3) y (4), se puede comprobar que $\vec{p} * \vec{b} = 0$, para todo $\vec{p} \in N_\alpha$ y $\vec{b} \in N_{\bar{\alpha}}$. Más aún, de estas mismas ecuaciones se puede concluir, en general, que $N_\alpha \cap N_{\bar{\alpha}} = \{0\}$.

Como una consecuencia inmediata de la propiedad anterior tenemos el siguiente:

Corolario 1.22. *Sean m, S y α como antes, y sea $\beta \subseteq \alpha$, entonces:*

$$(\mu^\beta)_{b+p} = (\mu^\beta)_b \ \ \forall\, b \in N_{\bar{\alpha}} \ \textit{y todo}\ p \in N_\alpha \ .$$

Ejemplo 1.23. Para $m = 5$ con $\alpha = \{1, 2, 5\}$ y $\beta = \{2, 5\}$. De los conjuntos N_α y $N_{\bar{\alpha}}$, descritos en (3) y (4) respectivamente, vemos que $N_\alpha = \{0, 2, 4, 6\}$ y $N_{\bar{\alpha}} = \{0, 1, 8, 9, 16, 17, 24, 25\}$. Si por ejemplo, $b = 17$ y $p = 2$, entonces (recuerde que la representación binaria de 17 es $(1, 0, 0, 0, 1)$, mientras que para 19 es $(1, 0, 0, 1, 1)$):

$$(\mu^\beta)_{19} = v_2 v_5(19) = 0 \cdot 1 = v_2 v_5(17) = (\mu^\beta)_{17} \ .$$

Proposición 1.24. Con la notación anterior, sea $\alpha \in 2^S$ y $\beta \subseteq \alpha$, entonces:

$$\sum_{b \in N_{\bar{\alpha}}} (\mu^\beta)_b = \delta_\alpha(\beta) \ .$$

donde la suma, al lado izquierdo de la ecuación anterior, se toma en $GF(2)$ y la función δ es la delta de Kronecker (ver por ejemplo [1]).

Demostración: Dados α y β construimos el siguiente conjunto:

$$A = \{\vec{p} = (p_1, p_2, \ldots, p_m) \in N_{\bar{\alpha}} \mid p_i = 1 \quad \forall \ i \in \beta\} \ .$$

Por como hemos construido A, tenemos que $|A| = 2^{|\alpha \setminus \beta|}$ y $(\mu^\beta)_b = 1 \iff b \in A$ para toda b en $N_{\bar{\alpha}}$. Por tanto $\sum_{b \in N_{\bar{\alpha}}} (\mu^\beta)_b = \sum_{b \in A} (\mu^\beta)_b = \sum_{2^{|\alpha \setminus \beta|}} 1 = \delta_\alpha(\beta)$. □

Ejemplo 1.25. Para $m = 5$ con $\alpha = \{1, 2, 5\}$. Dado que $|N_{\bar{\alpha}}| = 2^{m-|\bar{\alpha}|} = 2^{m-|S \setminus \alpha|} = 2^{m-(m-|\alpha|)} = 2^{|\alpha|}$, entonces para este ejemplo tenemos que $|N_{\bar{\alpha}}| = 8$, lo cual se puede comprobar directamente de (4). Ahora si $\beta = \phi$ entonces $\sum_{b \in N_{\bar{\alpha}}} (1)_b = \sum_8 (1) = 0$. Si $\beta = \{2\} \subset \alpha$, tenemos que $A = \{(0,1,0,0,0), (0,1,0,0,1), (1,1,0,0,0), (1,1,0,0,1)\}$ (ver elementos de $N_{\bar{\alpha}}$ en (4)) y por tanto $\sum_{b \in N_{\bar{\alpha}}} (\mu^\beta)_b = \sum_4 (1) = 0$. Si $\beta = \{2, 5\} \subset \alpha$, tenemos que $A = \{(0,1,0,0,1), (1,1,0,0,1)\}$ y $\sum_{b \in N_{\bar{\alpha}}} (\mu^\beta)_b = \sum_2 (1) = 0$. Por último, si $\beta = \alpha$, tenemos que $A = \{(1,1,0,0,1)\}$ y por tanto $\sum_{b \in N_{\bar{\alpha}}} (\mu^\beta)_b = \sum_1 (1) = 1$.

El siguiente lema será de importancia en la justificación del algoritmo de decodificación para los códigos de Reed-Muller.

Lema 1.26. *Sean α y β en 2^S, y sean $p \in N_\alpha$ y $b \in N_{\bar{\alpha}}$. Si $\beta_1 = \alpha \cap \beta$ y $\beta_2 = \bar{\alpha} \cap \beta$, entonces:*

$$(\mu^\beta)_{b+p} = (\mu^{\beta_1})_b \cdot (\mu^{\beta_2})_p \ .$$

Demostración: Claramente $\beta_1 \cup \beta_2 = \beta$ y $\beta_1 \cap \beta_2 = \phi$, por tanto $(\mu^\beta)_{b+p} = (\mu^{\beta_1 \cup \beta_2})_{b+p} = (\mu^{\beta_1} \cdot \mu^{\beta_2})_{b+p} = (\mu^{\beta_1})_{b+p} \cdot (\mu^{\beta_2})_{b+p}$. Ahora bien, dado que $\beta_1 \subseteq \alpha$ y $\beta_2 \subseteq \bar{\alpha}$, entonces aplicando el Corolario 1.22 tenemos que $(\mu^{\beta_1})_{b+p} = (\mu^{\beta_1})_b$ y $(\mu^{\beta_2})_{b+p} = (\mu^{\beta_2})_p$, con lo cual se llega al resultado deseado. □

Ejemplo 1.27. Para $m = 5$ con $\alpha = \{1, 2, 5\}$ y $\beta = \{2, 3, 5\}$. Tenemos entonces que $\bar{\alpha} = \{3, 4\}$ y por tanto $\beta_1 = \{2, 5\}$ y $\beta_2 = \{3\}$. Si por ejemplo, $b = 17$ y $p = 2$, entonces $(\mu^\beta)_{19} = v_2 v_3 v_5(19) = 0 \cdot 0 \cdot 1 = 0$. Por otro lado, $(\mu^{\beta_1})_{17} \cdot (\mu^{\beta_2})_2 = (v_2 v_5(17)) \cdot v_3(2) = (0 \cdot 1) \cdot 0 = 0$.

2. Códigos de Reed-Muller

Los códigos de Reed-Muller fueron presentados inicialmente por I. S. Reed [3] y por D. E. Muller [2]. Este tipo de códigos pertenecen a la familia de los códigos lineales los cuales se caracterizan por su sencillez tanto en su descripción como en el proceso de codificación y decodificación [1].

Definición 2.1. Sea m como antes. Para un entero r con $0 \leq r \leq m$, definimos el código binario de Reed-Muller de orden r en m variables, $\mathcal{R}(r,m)$, como el conjunto de todas las funciones booleanas (o equivalentemente el conjunto de las tablas de verdad correspondientes) en $\mathcal{B}_m$ de grado a lo más r. Es decir:

$$\mathcal{R}(r,m) = \{f \in \mathcal{B}_m \mid \text{grado}(f) \leq r\} .$$

Dos propiedades importantes de los códigos de Reed-Muller, las cuales se siguen directamente de la definición, son:

Propiedades 2.2. Sean m y r enteros positivos con $r \leq m$, entonces:

$$\mathcal{R}(0,m) \subset \mathcal{R}(1,m) \subset \mathcal{R}(2,m) \subset \cdots \subset \mathcal{R}(m,m) = \mathcal{B}_m \ , \ \text{ y además,}$$

$$\mathcal{R}(r,m) \subset \mathcal{R}(r,m+1) .$$

Debido a la longitud de las tablas de verdad de las funciones en $\mathcal{B}_m$, se sigue que la longitud de código para $\mathcal{R}(r,m)$ es 2^m. Más aún, claramente $\mathcal{R}(r,m)$ es un subespacio vectorial de $\mathcal{B}_m$ y por tanto $\mathcal{R}(r,m)$ es un código lineal. Una base para $\mathcal{R}(r,m)$ es:

$$B = \{\mu^\alpha \mid \alpha \in 2^S \text{ y } |\alpha| \leq r\} , \tag{5}$$

y por consiguiente la dimensión k, de $\mathcal{R}(r,m)$ es:

$$k = dim(\mathcal{R}(r,m)) = \binom{m}{0} + \binom{m}{1} + \cdots + \binom{m}{r} . \tag{6}$$

Observe cómo el conjunto de tablas de verdad de las funciones booleanas en (5), dan lugar a una matriz binaria de orden $k \times 2^m$, la cual, en sí, puede ser considerada como la matriz generadora del código $\mathcal{R}(r,m)$.

Ejemplo 2.3. Para $m = 3$ y $r = 2$, entonces una base B para R(2,3) es:

$$B = \{\mu^\phi, \mu^{\{1\}}, \mu^{\{2\}}, \mu^{\{3\}}, \mu^{\{1,2\}}, \mu^{\{1,3\}}, \mu^{\{2,3\}}\} = \{1, v_1, v_2, v_3, v_1v_2, v_1v_3, v_2v_3\} ,$$

y si empleamos las tablas de verdad de los elementos de esta base,

tenemos que:

$$B = \left\{ \begin{array}{l} (1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1), \\ (0,\ 0,\ 0,\ 0,\ 1,\ 1,\ 1,\ 1), \\ (0,\ 0,\ 1,\ 1,\ 0,\ 0,\ 1,\ 1), \\ (0,\ 1,\ 0,\ 1,\ 0,\ 1,\ 0,\ 1), \\ (0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 1,\ 1), \\ (0,\ 0,\ 0,\ 0,\ 0,\ 1,\ 0,\ 1), \\ (0,\ 0,\ 0,\ 1,\ 0,\ 0,\ 0,\ 1) \end{array} \right\} \begin{array}{l} \longleftrightarrow \\ \longleftrightarrow \\ \longleftrightarrow \\ \longleftrightarrow \\ \longleftrightarrow \\ \longleftrightarrow \\ \longleftrightarrow \end{array} \left\{ \begin{array}{l} 1, \\ v_1, \\ v_2, \\ v_3, \\ v_1v_2, \\ v_1v_3, \\ v_2v_3 \end{array} \right\} .$$

La matriz generadora del código $\mathcal{R}(2,3)$, es la matriz binaria que aparece a la izquierda de la relación anterior.

Ya conocemos en general cual es la longitud y la dimensión de un código de Reed-Muller de orden r, sin embargo, ¿cuál será su distancia mínima? Aunque ahora no se ve una solución directa a esta pregunta, sí podemos responder a ésta en casos muy particulares. Por ejemplo, si $r = 0$ entonces $\mathcal{R}(r,m)$ es el código de repetición de longitud 2^m y por tanto su distancia mínima es 2^m. Por otro lado, si $r = m$ entonces $\mathcal{R}(r,m) = \mathcal{B}_m$ y por tanto su distancia mínima es 1. Para dar una respuesta general a esta pregunta, será necesario presentar algunos resultados preliminares.

Lema 2.4. *Sean m y r enteros positivos con $r < m$, entonces:*

$$\sum_{i=0}^{r} \binom{m}{i} + \sum_{i=0}^{r+1} \binom{m}{i} = \sum_{i=0}^{r+1} \binom{m+1}{i} .$$

Demostración:

$$\begin{aligned} \sum_{i=0}^{r} \binom{m}{i} + \sum_{i=0}^{r+1} \binom{m}{i} &= \binom{m}{0} + \sum_{i=1}^{r+1} \left[\binom{m}{i} + \binom{m}{i-1} \right] \\ &= \binom{m+1}{0} + \sum_{i=1}^{r+1} \binom{m+1}{i} = \sum_{i=0}^{r+1} \binom{m+1}{i} . \end{aligned}$$

□

Teorema 2.5. *Sean m y r enteros positivos con $r < m$, entonces:*

$$\mathcal{R}(r+1, m+1) = \{ |u|u+v| \ \mid \ u \in \mathcal{R}(r+1,m),\ v \in \mathcal{R}(r,m) \} . \quad (7)$$

Demostración: Sea $f \in \mathcal{R}(r+1, m+1)$, entonces existen funciones booleanas g y h tales que:

$$f(v_1, v_2, ..., v_{m+1}) = g(v_2, v_3, ..., v_{m+1}) + v_1 h(v_2, v_3, ..., v_{m+1}) ,$$

donde grado$(g) \leq r+1$, grado$(h) \leq r$ y por tanto $g \in \mathcal{R}(r+1,m) \subset \mathcal{R}(r+1,m+1)$ y $h \in \mathcal{R}(r,m) \subset \mathcal{R}(r+1,m+1)$. Si $T_v(g)$ y $T_v(h)$ son las tablas de verdad de g y h, vistas éstas como elementos de $\mathcal{B}_m$, y si $\vec{0}$ es la tabla de verdad de la función booleana constante igual a cero en $\mathcal{B}_m$, entonces las tablas de verdad, $T_v(g)'$ y $T_v(v_1h)'$, de g y v_1h vistas ahora éstas como elementos de $\mathcal{B}_{m+1}$, estarán dadas por:

$$T_v(g)' = |T_v(g)|T_v(g)| \text{ y } T_v(v_1h)' = |\vec{0}|T_v(h)| \ ,$$

y por tanto, si $T_v(f)'$ es la tabla de verdad de f en $\mathcal{B}_{m+1}$, tenemos que:

$$T_v(f)' = T_v(g)' + T_v(v_1h)' = |T_v(g)|T_v(g) + T_v(h)| \ ,$$

lo cual muestra que $f \in \{|u|u+v| \mid u \in \mathcal{R}(r+1,m),\ v \in \mathcal{R}(r,m)\}$, y de esta manera se ha probado que el conjunto del lado izquierdo de (7), está totalmente contenido en su contraparte del lado derecho. Ahora bien, ambos conjuntos en (7) son espacios vectoriales, donde claramente la dimensión del conjunto del lado derecho es:

$$\sum_{i=0}^{r} \left(\begin{array}{c} m \\ i \end{array} \right) + \sum_{i=0}^{r+1} \left(\begin{array}{c} m \\ i \end{array} \right) = \sum_{i=0}^{r+1} \left(\begin{array}{c} m+1 \\ i \end{array} \right) = dim(\mathcal{R}(r+1,m+1)) \ .$$

□

Ejemplo 2.6. Con $r = 2$, $m = 3$ y si $f \in \mathcal{R}(3,4)$ está dada por $f(v_1,v_2,v_3,v_4) = 1+v_4+v_1v_2+v_1v_2v_4+v_2v_3v_4 = 1+v_4+v_2v_3v_4+v_1(v_2+v_4v_2)$, entonces, $g = 1+v_4+v_2v_3v_4 \in \mathcal{R}(3,3)$ y $h = v_2+v_4v_2 \in \mathcal{R}(2,3)$. $T_v(g) = (10101011)$, $T_v(h) = (00001010)$ y:

$$T_v(f)' = (1010101110100001) = |\ (10101011)\ |\ (10101011)+(00001010)\ | \ .$$

El siguiente teorema clásico [1, Teorema 33, Capítulo 2, p. 76], nos da información acerca de los parámetros de un código binario, el cual ha sido obtenido a través de la construcción $|u|u+v|$.

Teorema 2.7. *Sean $\mathcal{C}_1$ y $\mathcal{C}_2$ dos códigos binarios (no necesariamente lineales) con parámetros (n, M_1, d_1) y (n, M_2, d_2) respectivamente. Si $\mathcal{C}_3$ es la construcción $|u|u+v|$ de $\mathcal{C}_1$ y $\mathcal{C}_2$, es decir:*

$$\mathcal{C}_3 = \{|u|u+v| \ \mid \ u \in \mathcal{C}_1,\ v \in \mathcal{C}_2\} \ ,$$

entonces $\mathcal{C}_3$ tiene parámetros $(2n, M_1M_2, d_3 = min\{2d_1, d_2\})$.

Con ayuda del teorema anterior, podemos ahora determinar explícitamente la distancia mínima para los código de Reed-Muller.

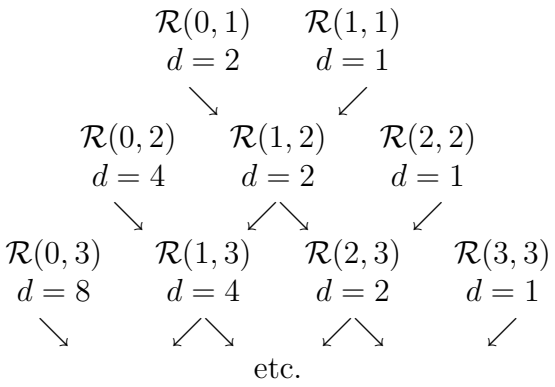

Figura 1: Obtención de las distancias mínimas de los códigos de Reed-Muller de cualquier orden, a través de la aplicación recursiva de los Teoremas 2.5 y 2.7.

Teorema 2.8. *El código $\mathcal{R}(r,m)$ tiene distancia mínima 2^{m-r}.*

Demostración: Ya se hizo la observación de que los códigos de la forma $\mathcal{R}(0,m)$ y $\mathcal{R}(m,m)$ tienen distancia mínima 2^m y 1, respectivamente. Ahora bien, según el Teorema 2.5, el código $\mathcal{R}(1,2)$ puede ser construido en términos de los códigos $\mathcal{R}(0,1)$ y $\mathcal{R}(1,1)$, los cuales tienen como distancia mínima 2 y 1, respectivamente. Aplicando ahora el Teorema 2.7, a estos dos códigos, llegamos entonces a la conclusión de que $\mathcal{R}(1,2)$ debe tener como distancia mínima 2. Como se puede apreciar de la Figura 1, la aplicación recursiva del procedimiento anterior permite obtener el resultado deseado. □

Ejemplo 2.9.

$\mathcal{R}(2,4)= \mathcal{C}[16,11,4]$ ⟵ Código binario extendido de Hamming de orden 4.
$\mathcal{R}(1,5)= \mathcal{C}[32,6,16]$ ⟵ Usado por el Mariner 9.
$\mathcal{R}(2,5)= \mathcal{C}[32,16,8]$ ⟵ ¡Bonitos parámetros!, ideal para una implementación en software.

3. Algoritmo de Codificación

Dado que $\mathcal{R}(r,m)$ es un código lineal, el esquema de codificación será a través de la matriz generadora, cuya dimensión es de $k \times 2^m$, donde el número de renglones k está dado según (6). Ahora bien, para cada $\beta \in 2^S$ con $|\beta| \leq r$ denotaremos por a_β al *bit información* dentro

del siguiente vector binario de longitud k:

$$A=(a_\phi, a_{\{1\}}, a_{\{2\}}, \ldots, a_{\{m\}}, a_{\{1,2\}}, a_{\{1,3\}}, \ldots, a_{\{1,m\}}, \ldots, a_{\{m-r+1,m-r+2,\ldots,m\}}) \ . \tag{8}$$

Ejemplo 3.1. Si $m = 4$ y $r = 3$, entonces $k = 15$ y:

$$\begin{aligned} A \ = \ & (a_\phi, a_{\{1\}}, a_{\{2\}}, a_{\{3\}}, a_{\{4\}}, a_{\{1,2\}}, a_{\{1,3\}}, a_{\{1,4\}}, a_{\{2,3\}}, a_{\{2,4\}}, a_{\{3,4\}}, \\ & a_{\{1,2,3\}}, a_{\{1,2,4\}}, a_{\{1,3,4\}}, a_{\{2,3,4\}}) \ . \end{aligned}$$

En este esquema de codificación, el vector A, dado por (8), será considerado como el vector binario de información a codificar. Por tanto, la palabra de código f que codifica al vector A en $\mathcal{R}(r, m)$, estará dado por:

$$f = \sum_{|\beta| \leq r} a_\beta \mu^\beta \ ,$$

o de manera equivalente, en términos de la tabla de verdad de f:

$$f(b) = \sum_{|\beta| \leq r} a_\beta (\mu^\beta)_b \quad \forall \ b \in \{0, 1, \ldots, 2^m - 1\} \ . \tag{9}$$

4. Algoritmo de Decodificación

Uno de los algoritmos de decodificación más sencillos para códigos lineales, es la decodificación a través de síndromes. Para el caso de los códigos de Reed-Muller esta estrategia de decodificación es, en general, poco conveniente. Por ejemplo, para el código $\mathcal{R}(1, 5)$, que se empleó para las fotos de Marte, el número de síndromes necesarios para la decodificación bajo este esquema es ¡2^{26}!

Una alternativa de decodificación es el llamado *Reed-algorithm*, el cual basa su principio de decodificación en la llamada lógica de mayoría o decodificación por votación. La idea es la siguiente; dado el vector de información A, éste se codifica, según (9), para dar lugar a la palabra de código $T_v(f)$, la cual de no sufrir alteración alguna, debería permitir la recuperación del vector de información A, ¿pero cómo hacer esto? El siguiente teorema, no sólo da la respuesta a esta pregunta, sino que también nos da la pista para el algoritmo de decodificación.

Teorema 4.1. *Sea $\mathcal{R}(r, m)$ el código de Reed-Muller de orden r y sean A y $T_v(f)$ según (8) y (9), respectivamente, entonces:*

$$a_\alpha = \sum_{b \in N_{\bar{\alpha}}} f(b+p) \quad con \;\; p \in N_\alpha \;\; y \;\; |\alpha| = r \; . \tag{10}$$

Antes de pasar a la demostración del teorema, es importante hacer las siguientes dos observaciones:

- Debe notarse que el teorema no recupera todos los a_α, sino sólo aquellos con $|\alpha| = r$. Sin embargo, si se recuperan éstos, entonces observe que:

  $$f' = f - \sum_{|\beta|=r} a_\beta \mu^\beta \in \mathcal{R}(r-1, m) \; ,$$

  y por tanto, podemos re-aplicar el teorema para f', con lo cual se recuperarán ahora los a_α con $|\alpha| = r - 1$. Aplicando recursivamente esta idea podremos entonces recuperar sucesivamente todos los a_α, hasta llegar a a_ϕ.

- Observe que por cada $p \in N_\alpha$, se tiene una ecuación que determina a_α, en (10), y por tanto, el número de estas ecuaciones es $|N_\alpha| = 2^{m-|\alpha|}$. De esta manera, el teorema anterior nos dice que para un esquema de decodificación de $T_v(f)$, bastaría con una sola de estas ecuaciones para recuperar el *bit* de información a_α, siempre y cuando $T_v(f)$ no haya sufrido alteraciones. Por el contrario, si algunos *bits* de $T_v(f)$ han sido alterados, entonces algunos de los valores para estas ecuaciones estarán mal y otros bien. Si es posible estimar que el número de *bits* en error, en $T_v(f)$, no es muy grande, entonces, también, se podrá estimar que el número de ecuaciones con un valor correcto será mayor, que aquellas con un valor incorrecto. Esta es precisamente la idea de la decodificación por lógica de mayoría o decodificación por votación. Es decir, cada uno de los valores de estas ecuaciones aportará un voto para los dos posibles valores de a_α, a saber cero o uno. De estas votaciones, se identificará aquel valor que reciba la mayor votación, el cual será considerado como el valor de decodificación para a_α.

Demostración: Sea $\alpha \in 2^S$ con $|\alpha| = r$. Si para una $p \in N_\alpha$, aplicamos la sumatoria: $\sum_{b \in N_{\bar{\alpha}}}$, en ambos lados de (9), se tiene que:

$$\sum_{b \in N_{\bar{\alpha}}} f(b+p) = \sum_{|\beta| \leq r} a_\beta \sum_{b \in N_{\bar{\alpha}}} (\mu^\beta)_{b+p} \quad \text{con} \;\; p \in N_\alpha \text{ y } \;\; |\alpha| = r \; .$$

Para cada $|\beta| \leq r$, sean $\beta_1 = \alpha \cap \beta$ y $\beta_2 = \bar{\alpha} \cap \beta$, entonces aplicando la Lema 1.26 a la parte derecha de la igualdad anterior, tenemos que:

$$\sum_{b \in N_{\bar{\alpha}}} f(b+p) = \sum_{|\beta| \leq r} a_\beta (\mu^{\beta_2})_p \sum_{b \in N_{\bar{\alpha}}} (\mu^{\beta_1})_b \ ,$$

pero como $\beta_1 \subseteq \alpha$, entonces podemos aplicar la Proposición 1.24:

$$\sum_{b \in N_{\bar{\alpha}}} f(b+p) = \sum_{|\beta| \leq r} a_\beta (\mu^{\beta_2})_p \delta_\alpha(\beta_1) \ .$$

Ahora bien, dado que $|\beta| \leq r$ y $|\alpha| = r$, entonces si $\beta_1 = \alpha$, tenemos necesariamente que $\beta = \alpha$, $\beta_2 = \phi$ y $\mu^{\beta_2} = 1$, con lo cual concluimos entonces que $\sum_{b \in N_{\bar{\alpha}}} f(b+p) = a_\alpha$. □

Ejemplo 4.2. Para $m = 4$ y $r = 2$, tenemos que si $\alpha = \{1, 2\}$, entonces $N_\alpha = \{0, 1, 2, 3\}$ y $N_{\bar{\alpha}} = \{0, 4, 8, 12\}$, y por tanto las cuatro ecuaciones que determinan a $a_{\{1,2\}}$ son:

$$\begin{array}{rllll} a_{\{1,2\}} & = f(0) & + f(4) & + f(8) \quad + f(12) & \longleftarrow \quad \text{Voto 1} \\ & = f(1) & + f(5) & + f(9) \quad + f(13) & \longleftarrow \quad \text{Voto 2} \\ & = f(2) & + f(6) & + f(10) \quad + f(14) & \longleftarrow \quad \text{Voto 3} \\ & = f(3) & + f(7) & + f(11) \quad + f(15) & \longleftarrow \quad \text{Voto 4 .} \end{array} \tag{11}$$

Si ahora $\alpha = \{1, 3\}$, entonces $N_\alpha = \{0, 1, 4, 5\}$ y $N_{\bar{\alpha}} = \{0, 2, 8, 10\}$, y por tanto las cuatro ecuaciones que determinan a $a_{\{1,3\}}$ son:

$$\begin{array}{rllll} a_{\{1,3\}} & = f(0) & + f(2) & + f(8) \quad + f(10) & \longleftarrow \quad \text{Voto 1} \\ & = f(1) & + f(3) & + f(9) \quad + f(11) & \longleftarrow \quad \text{Voto 2} \\ & = f(4) & + f(6) & + f(12) \quad + f(14) & \longleftarrow \quad \text{Voto 3} \\ & = f(5) & + f(7) & + f(13) \quad + f(15) & \longleftarrow \quad \text{Voto 4 .} \end{array} \tag{12}$$

De manera análoga, se determinan los conjuntos de ecuaciones que determinarán a $a_{\{1,4\}}$, $a_{\{2,3\}}$, $a_{\{2,4\}}$ y $a_{\{3,4\}}$.

Según el Teorema 2.8, el código, $\mathcal{R}(2, 4)$, tiene distancia mínima 4, y por tanto, a través de éste, se puede diseñar un código capaz de corregir un error y detectar hasta dos errores. De esta manera, observe que si se presenta un sólo error en $T_v(f)$, entonces tres de las cuatro votaciones que determinarán a a_α, con $|\alpha| = 2$ (ver como ejemplo los conjuntos de ecuaciones para (11) y (12)), tendrán un valor igual, el cual será el

valor a decodificar para a_α. Una vez estimados todos los valores para a_α con $|\alpha| = 2$, se procederá a estimar ahora los a_α con $|\alpha| = 1$, para lo cual se realiza, primero, el cálculo de la siguiente función booleana:

$$f' = f - \sum_{|\alpha|=2} a_\alpha \mu^\alpha \, .$$

Si las estimaciones para a_α, con $|\alpha| = 2$, fueron las correctas, entonces $T_v(f')$ deberá corresponder a una palabra de código en $\mathcal{R}(1,4)$, la cual a lo sumo tiene un bit en error. Pero $\mathcal{R}(1,4)$ es un $\mathcal{C}[16,5,8]$, el cual es capaz de corregir hasta 3 errores. Por tanto, si mantenemos el supuesto de que $T_v(f)$ a sufrido a lo sumo un error, entonces con mayor razón se podrán estimar todos los a_α con $|\alpha| = 1$ usando ahora el código $\mathcal{R}(1,4)$. Por ejemplo, para estimar $a_{\{1\}}$ tendríamos que $N_\alpha = \{0,1,2,3,4,5,6,7\}$ y $N_{\bar{\alpha}} = \{0,8\}$, y por tanto las ocho ecuaciones o votaciones que determinan a $a_{\{1\}}$ serían:

$$\begin{array}{llllll|llllll}
a_{\{1\}} & = f'(0) & + f'(8) & \leftarrow & \text{Voto 1} & & a_{\{1\}} & = f'(4) & + f'(12) & \leftarrow & \text{Voto 5} \\
 & = f'(1) & + f'(9) & \leftarrow & \text{Voto 2} & & & = f'(5) & + f'(13) & \leftarrow & \text{Voto 6} \\
 & = f'(2) & + f'(10) & \leftarrow & \text{Voto 3} & & & = f'(6) & + f'(14) & \leftarrow & \text{Voto 7} \\
 & = f'(3) & + f'(11) & \leftarrow & \text{Voto 4} & & & = f'(7) & + f'(15) & \leftarrow & \text{Voto 8} \, .
\end{array}$$

De haberse presentado a lo sumo un error en $T_v(f)$, y por tanto también a lo sumo un sólo error en $T_v(f')$, entonces por lo menos 7 de las 8 votaciones anteriores tendrán el mismo valor, el cual deberá ser la decodificación para $a_{\{1\}}$. De manera análoga, se estimarían los valores para $a_{\{2\}}$, $a_{\{3\}}$ y $a_{\{4\}}$. Ya obtenidos todos los a_α con $|\alpha| = 1$, se calcularía la función booleana:

$$f'' = f' - \sum_{|\alpha|=1} a_\alpha \mu^\alpha \, ,$$

la cual, de no presentarse errores, debería ser un elemento en $\mathcal{R}(0,4)$. Pero $\mathcal{R}(0,4)$ es el código de repetición de longitud 16, y por tanto, a_ϕ podrá ahora ser estimado en términos de $T_v(f'')$. Es decir, para esta última estimación se tendría que $N_\alpha = \{0,1,2,\ldots,15\}$ y $N_{\bar{\alpha}} = \{0\}$ y por consiguiente las 16 ecuaciones que determinarán a a_ϕ son:

$$a_\phi = f''(i) \quad \text{para } i \in \{0,1,2,\ldots,15\} \, ,$$

cuyos valores deberán coincidir en por lo menos 15 votaciones.

Apéndice

Con el objetivo de ejemplificar los conceptos vistos en las secciones anteriores, a continuación se presenta la implementación en lenguaje "C", de los algoritmos de codificación y decodificación asociados al

código de Reed-Muller de orden 2 en 5 variables ($\mathcal{R}(2,5)$). Tal implementación corresponde a tres programas en lenguaje "C"; *RMCodifica.c*, *Canal.c* y *RMDecodifica.c*. El primer y tercer programa corresponden, respectivamente, a las implementaciones de los algoritmos de codificación y decodificación vistos en este trabajo para el caso $\mathcal{R}(2,5)$. El segundo programa, implementa la simulación de un canal ruidoso. La idea es que estos tres programas se ejecuten como procesos independientes pero interconectados a través de un *pipe* ("|"), de tal manera que la salida del proceso de codificación (*ie*, el programa *RMCodifica.c*) sea la entrada al proceso canal (*ie*, el programa *Canal.c*), cuya salida es a su vez la entrada al proceso de decodificación (*ie*, el programa *RMDecodifica.c*).

Cada uno de estos procesos requiere de un parámetro para su ejecución. Para el caso de la codificación se requiere de un nombre de archivo, el cual será usado para obtener una muestra de información a codificar (se hace la aclaración, que el archivo correspondiente no es modificado por este proceso). El proceso canal requiere de un entero $n > 0$, por medio del cual se determina que por cada n *bits* que "pasan" por el canal, uno, en promedio, es alterado. Para el caso del proceso de decodificación, este requiere como parámetro de el mismo nombre de archivo usado para el proceso de codificación. La razón de lo anterior se debe a que este último proceso no sólo realiza la tarea de decodificación, sino que además re-codifica la información original con el objetivo de auto evaluar su capacidad de corrección ante los errores introducidos por el canal. Con el objeto de ilustrar lo anterior, se presenta a continuación un ejemplo de ejecución de estos tres procesos:

```
% RMCodifica light.gif | Canal 200 | RMDecodifica light.gif
                                Wt(X-X')     Wt(A-A^)
        Errores de peso = 0        59762        70229
        Errores de peso = 1         9668            0
        Errores de peso = 2          753            1
        Errores de peso = 3           46            0
        Errores de peso = 4            2            0
        Errores de peso = 6            0            1
   P = 11320/2247392 = 0.00503694964573
   Psymb = 8/1123696 = 0.00000711936309
```

En este ejemplo se codifica la información contenida en el archivo `light.gif` (una imagen *gif* de 140,462 *bytes*). Esta información codificada es entregada al proceso `Canal` el cual, como puede verse, alterará en promedio 1 de cada $n = 200$ bits que "pasen" por él. El proceso de decodificación (`RMDecodifica`), recibe esta información y

con la ayuda de la codificación original (*ie*, la misma codificación que el proceso `RMCodifica` genera con el archivo `light.gif`), procede a su autoevaluación. Para la presentación de esta autoevaluación, se emplea la siguiente notación: `X` denota una palabra de código en $\mathcal{R}(2,5)$ (*ie*, 32 *bits*). `X'` denota a la misma palabra de código, después de que ésta ha pasado por el canal (proceso `Canal`). `Wt(X-X')` denota en número de posiciones en donde varian `X` y `X'`. De esta manera, la secuencia de números debajo de etiqueta `Wt(X-X')` contabiliza el número de palabras de código `X` que fueron alteradas por el canal en r posiciones (`Errores de peso =` r). Por otro lado, `A` denota a los 16 *bits* que codifican `X`, mientras que `A^` denota la estimación de estos 16 *bits* después de la decodificación. Así pues, la secuencia de números debajo de `Wt(A-A^)` contabiliza el número de veces en que `A` y `A^` difirieron en r posiciones (*ie*, para $r \geq 1$, esta columna contabiliza los errores después de la decodificación). Por último, `P` es una estimación de la probabilidad de que un *bit* al pasar por el canal sea alterado por éste (*ie*, `P` debe ser aproximadamente igual a $1/n$) y `Psymb` denota la probabilidad de que un *bit* de información (ver Sección 3) esté en error después de la decodificación (claramente es de esperarse que `P` $>$ `Psymb`).

Lo que resta de este apéndice, es el código fuente de los programas *RMCodifica.c*, *Canal.c* y *RMDecodifica.c*.

RMCodifica.h

```
/* Palabras de codigo */
unsigned int PC1[16] = { /* para 1, v1, v2 y v3 */
   0x00000000,0xffffffff,0xffff0000,0x0000ffff,
   0xff00ff00,0x00ff00ff,0x00ffff00,0xff0000ff,
   0xf0f0f0f0,0x0f0f0f0f,0x0f0ff0f0,0xf0f00f0f,
   0x0ff00ff0,0xf00ff00f,0xf00f0ff0,0x0ff0f00f};
unsigned int PC2[16] = { /* para v4, v5, v12 y v13 */
   0x00000000,0xcccccccc,0xaaaaaaaa,0x66666666,
   0xff000000,0x33cccccc,0x55aaaaaa,0x99666666,
   0xf0f00000,0x3c3ccccc,0x5a5aaaaa,0x96966666,
   0x0ff00000,0xc33ccccc,0xa55aaaaa,0x69966666};
unsigned int PC3[16] = { /* para v14, v15, v23 y v24 */
   0x00000000,0xcccc0000,0xaaaa0000,0x66660000,
   0xf000f000,0x3cccf000,0x5aaaf000,0x9666f000,
   0xcc00cc00,0x00cccc00,0x66aacc00,0xaa66cc00,
   0x3c003c00,0xf0cc3c00,0x96aa3c00,0x5a663c00};
unsigned int PC4[16] = { /* para v25, v34, v35 y v45 */
   0x00000000,0xaa00aa00,0xc0c0c0c0,0x6ac06ac0,
```

```
   0xa0a0a0a0,0x0aa00aa0,0x60606060,0xca60ca60,
   0x88888888,0x22882288,0x48484848,0xe248e248,
   0x28282828,0x82288228,0xe8e8e8e8,0x42e842e8};
```

RMCodifica.c

```
#include <stdio.h>
#include ``RMCodifica.h"

main(int argc, char* argv[])
{
   FILE *fp;
   int c1, c2;
   unsigned int x;

   if((fp = fopen(argv[1],"r")) == NULL) {
      printf(``Error en el archivo%s\n",argv[1]); exit(1);
   }
   while((c1=fgetc(fp))!=EOF) {
      if((c2=fgetc(fp))==EOF) break;
      x = PC1[c1&15]^PC2[(c1>>4)&15]^
          PC3[c2&15]^PC4[(c2>>4)&15];
      putchar(x&255);          putchar((x>>8)&255);
      putchar((x>>16)&255);    putchar((x>>24)&255);
   }
   fclose(fp);
}
```

Canal.c

```
#include <stdio.h>
#include <stdlib.h>
#define N 8
main(int argc, char* argv[])
{
   int i, c, invp, e;

   srand((int) getpid());
   invp = atoi(argv[1]);
   while((c=getchar())!=EOF) {
      e = 0;
      for(i=0;i<N;i++) if(!(rand()%invp)) e ^= (1<<i);
      putchar(c^e);
   }
}
```

RMDecodifica.c

```
#include <stdio.h>
#include ‘‘RMCodifica.h"
unsigned int Aq[320] = { /* Coeficientes cuadraticos */
   0, 8,16,24, 1, 9,17,25, 2,10,18,26, 3,11,19,27,
   4,12,20,28, 5,13,21,29, 6,14,22,30, 7,15,23,31,
   0, 4,16,20, 1, 5,17,21, 2, 6,18,22, 3, 7,19,23,
   8,12,24,28, 9,13,25,29,10,14,26,30,11,15,27,31,
   0, 2,16,18, 1, 3,17,19, 4, 6,20,22, 5, 7,21,23,
   8,10,24,26, 9,11,25,27,12,14,28,30,13,15,29,31,
   0, 1,16,17, 2, 3,18,19, 4, 5,20,21, 6, 7,22,23,
   8, 9,24,25,10,11,26,27,12,13,28,29,14,15,30,31,
   0, 4, 8,12, 1, 5, 9,13, 2, 6,10,14, 3, 7,11,15,
   16,20,24,28,17,21,25,29,18,22,26,30,19,23,27,31,
   0, 2, 8,10, 1, 3, 9,11, 4, 6,12,14, 5, 7,13,15,
   16,18,24,26,17,19,25,27,20,22,28,30,21,23,29,31,
   0, 1, 8, 9, 2, 3,10,11, 4, 5,12,13, 6, 7,14,15,
   16,17,24,25,18,19,26,27,20,21,28,29,22,23,30,31,
   0, 2, 4, 6, 1, 3, 5, 7, 8,10,12,14, 9,11,13,15,
   16,18,20,22,17,19,21,23,24,26,28,30,25,27,29,31,
   0, 1, 4, 5, 2, 3, 6, 7, 8, 9,12,13,10,11,14,15,
   16,17,20,21,18,19,22,23,24,25,28,29,26,27,30,31,
   0, 1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,13,14,15,
   16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31};
unsigned int Al[32*5] = { /* Coeficientes lineales */
   0,16, 1,17, 2,18, 3,19, 4,20, 5,21, 6,22, 7,23,
   8,24, 9,25,10,26,11,27,12,28,13,29,14,30,15,31,
   0, 8, 1, 9, 2,10, 3,11, 4,12, 5,13, 6,14, 7,15,
   16,24,17,25,18,26,19,27,20,28,21,29,22,30,23,31,
   0, 4, 1, 5, 2, 6, 3, 7, 8,12, 9,13,10,14,11,15,
   16,20,17,21,18,22,19,23,24,28,25,29,26,30,27,31,
   0, 2, 1, 3, 4, 6, 5, 7, 8,10, 9,11,12,14,13,15,
   16,18,17,19,20,22,21,23,24,26,25,27,28,30,29,31,
   0, 1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,13,14,15,
   16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31};

#define N 32
unsigned int DE1[N+1], DE2[N+1]; /* Dist. de errores */

main(int argc, char* argv[])
{
   FILE *fp;
   unsigned int i, a, A, c1, c2, d1, d2, d3, d4,
                x, y, tot0=0, tot1=0, tot2=0;
```

```
   if((fp = fopen(argv[1],"r")) == NULL) {
      printf(``Error en el archivo%s\n",argv[1]);
      exit(1);
   }
   memset(DE2,0,(N+1)*sizeof(unsigned int));
   while((c1=fgetc(fp))!=EOF) {
      if((c2=fgetc(fp))==EOF) break;
      y = PC1[c1&15]^PC2[(c1>>4)&15]^
          PC3[c2&15]^PC4[(c2>>4)&15];
      d1=getchar(); d2=getchar();
      d3=getchar(); d4=getchar();
      x = (d1&255)^((d2&255)<<8)^
          ((d3&255)<<16)^((d4&255)<<24);
      DE1[PH(x^y)]++;
      for(i=0,a=0;i<10;i++)
         a^= eval(x,&Aq[i*N],4) << (6+i);
      x^= PC2[(a>>4)&15]^PC3[(a>>8)&15]^
          PC4[(a>>12)&15];
      A = a;
      for(i=0,a=0;i<5;i++)
         a^= eval(x,&Al[i*N],2) << (1+i);
      x^= PC1[a&15]^PC2[(a>>4)&15];
      A^= a;
      if(PH(x)>(N/2)) A^= 1;
      DE2[PH(A^(c1^(c2<<8)))]++;
   }
   fclose(fp);
   printf(``\t\t\t\t Wt(X-X') Wt(A-A^)\n");
   for(i=0;i<N+1;i++) {
      if((DE1[i]==0) && (DE2[i]==0)) continue;
      tot0 += i*DE1[i];  tot1 += DE2[i];  tot2 += i*DE2[i];
      printf(``\t Errores de peso =%2d%14d%14d\n",
             i,DE1[i],DE2[i]);
   }
   printf(``P =%d/%d =%1.14f\n",tot0,tot1*32,
          ((float) tot0)/((float) tot1*32));
   printf(``Psymb =%d/%d =%1.14f\n",tot2,tot1*16,
          ((float) tot2)/((float) tot1*16));
}
```

```
eval(unsigned int x,A[],ns) /* Evalua las votaciones */
{
   unsigned int i, j, ne = N/(2*ns);
   unsigned char bit, vota[2] = {0,0};

   for(i=0;i<N;i+=ns) {
      for(j=0,bit=0;j<ns;j++) bit^= (x >> A[i+j])&1;
      vota[bit]++;
   }
   if(vota[0]>=ne) return(0);
   return(1);
}

PH(int x) /* Calcula el peso de Hamming */
{
   int i, p=0;

   for(i=0;i<N;i++,x >>= 1) p += (x&1);
   return(p);
}
```

Referencias

[1] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*, Amsterdam, The Netherlands: North-Holland, 1977.

[2] D. E. Muller, "Application of boolean algebra to swiching circuit design and error detection". *IEEE Trans. Computers*, Vol. 3, pp. 6-12, 1954.

[3] I. S. Reed, "A class of multiple-error-correcting codes and the decoding scheme". *IEEE Trans. Info. Theory*, Vol. 4, pp. 38-49, 1954.