

An Introduction to Golay Codes

Chris Winstead

February 2, 2000

1 Introduction

The purpose of this paper is to present the basic structure and properties of Golay codes to an audience who may have only limited experience with the theory of error control coding. A review of fundamental coding vocabulary is presented alongside a few of the concepts needed to understand the construction and properties of the binary Golay codes of length 23 and 24.

For simplicity, all concepts will be presented for the binary case only. The definitions provided will thus apply only to this special case of the general theory. In what follows, multiplication of two binary numbers may be thought of in the ordinary sense. An addition operation should be treated as an exclusive-OR.

A brief explanation of the trellis description of codes is then presented, including definitions for conventional and tail-biting trellises. Methods for constructing trellises from block codes using trellis products are also summarized.

The final sections present the hexacode and the Miracle Octad Generator (MOG), and explain their relationship to the (24, 12) Golay code. The MOG is then used to construct a minimal tail-biting trellis for the Golay code, as given by Calderbank, Forney, and Vardy in [3].

2 Review of Coding Terminology

It is desired to transmit a source message \vec{m} over a noisy channel in which one or more symbols may be corrupted. The received message \vec{r} may thus differ from the intended message by an error pattern \vec{e} , such that $\vec{r} = \vec{m} + \vec{e}$. The goal of Error Control Coding (ECC) is to introduce a controlled *redundancy* to transmitted messages, so that error patterns may be detected and corrected as often as possible by the receiver, restoring the intended messages.

The introduction of redundancy to the message is performed at the transmitting end by the encoder, which maps the intended message \vec{m} to a coded message, or codeword, \vec{c} . The codeword is then transmitted and decoded by the receiver. A simple approach to message encoding is provided by block codes.

2.1 Block Codes

An encoder for a block code C takes source messages of length k and produces codewords of length n . C thus represents a one-to-one mapping from a message vector-space V^k to a subspace of V^n .¹ Now C has 2^k codewords within a space of 2^n possible n -tuples. Because not all possible messages are valid codewords, we say that C has **redundancy**:

$$r = n - k$$

Another measure of a code's redundancy is its **rate**:

¹In general, C may take source messages of different lengths (e.g. if there are more codewords in C than there are vectors in V^k). But the codes in which we are interested use source messages of the same length k , so this case will be assumed throughout.

$$R = \frac{k}{n}$$

The Golay Code G_{24} , for example, takes source blocks of length 12 and produces codewords of length 24. It thus has rate $\frac{1}{2}$ and redundancy 12.

2.2 Description of Linear Codes

We say that a code C is **linear** if it forms a vector subspace over V^n . Thus it contains the zero-vector, and any linear combination of valid codewords results in a valid codeword. A linear code which takes source blocks of length k and has codewords of length n is labeled an (n, k) code. Because of their structure, linear codes may be analyzed using the tools of linear algebra and matrices. The first important matrix for characterizing C is its **generator matrix** \underline{G} :

$$\underline{G} = \begin{bmatrix} \vec{g}_0 \\ \vdots \\ \vec{g}_{k-1} \end{bmatrix}$$

where \vec{g}_i are the basis vectors for C . \underline{G} represents a mapping from the source vector-space to the codeword space. A message is thus encoded by:

$$\vec{m}\underline{G} = \vec{c}$$

A code may also be represented by its **parity-check matrix** \underline{H} . If C is a vector subspace generated by \underline{G} , then \underline{H} is defined as the set of basis vectors for the **dual-space**² of \underline{G} . If we say that C^\perp is the dual-space of C (or the dual code of C), then \underline{H} is the basis of C^\perp . Thus:

$$\underline{G} \times \underline{H}^T = \underline{0}$$

And, for any vector $\vec{c} \in V^n$, \vec{c} is a valid codeword if and only if:

$$\vec{c}\underline{H}^T = 0$$

2.3 Characteristics of Codewords

The **weight** of a codeword (or message, or error pattern) is defined as the number of non-zero coordinates in the codeword. The **Hamming Distance** between two codewords is the number of coordinates at which they differ. The **Minimum Distance** of a block code C is the minimum distance between all pairs of codewords in C . If d is the minimum distance of C , then C may sometimes be labeled as an (n, k, d) code.

For a linear code, the minimum distance of C is simply the weight of the lowest-weight codeword in C . Based upon these characteristics, the following may be said about a given linear block code C :

1. C can **detect** all errors of weight $\leq s$ if and only if the minimum distance $d(C) \geq s + 1$.
2. C can **correct** all errors of weight $\leq t$ if and only if $d(C) \geq 2t + 1$.

²The "Dual Space" (or "Orthogonal Space") is defined as follows: Let S be a subspace of a vector space V . The dual space S^\perp of S is the set of all vectors $\vec{v} \in V$ such that, for all $\vec{u} \in S$, $\vec{u} \cdot \vec{v} = 0$. S^\perp has the following characteristics:

- S and S^\perp are not disjoint ($\vec{0}$ is in both sets). S may furthermore be its own dual space.
- $\dim(S) + \dim(S^\perp) = \dim(V)$.
- S^\perp is also a vector subspace of V .

3. **Hamming Bound:** A binary code of length n which corrects t errors must have redundancy:

$$r \geq \log_2 \left\{ \sum_{j=0}^t \binom{n}{j} \right\}$$

For example, the Golay Code G_{24} has length 24 and minimum distance 8. It may thus detect error patterns of weight up to 7, correct up to 3 errors, and must have redundancy of at least 11.18.

A code which satisfies the Hamming Bound with equality (one which has minimum redundancy given its characteristics) is referred to as a **perfect code**.

3 Construction of the Golay Code

This section presents the Turyn construction of the Golay Code G_{24} , as described in [2, page 71]. The construction is based on the (7, 4) Hamming Code, which will be described first.

3.1 Hamming Codes

The (n, k) -Hamming Code (written Ham(r)) is defined as the code whose parity check matrix consists of all non-zero r -tuples. We will be primarily interested in Ham(3), whose parity-check matrix is:

$$\underline{H}_3 = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

The order of the columns is arbitrary. It may be easily verified that Ham(3) has the following characteristics:

- It has $n = 7$ and $k = 4$.
- It has a minimum distance of 3.
- All codewords other than $\vec{0}$ and $\vec{1}$ have weight 3 or 4.
- It is cyclic, and the complement of any codeword is also a codeword.

3.2 Extended Hamming Codes

Let H denote the Ham(3) code constructed above. Then let K denote the code obtained by reversing the bits in all the codewords of H (the columns of the generator matrix are set in reverse order). K is still a Ham(3) code with the properties designated above, but H and K share only $\vec{0}$ and $\vec{1}$.

Now we extend H and K to form H' and K' by adding a parity-check bit to each codeword, such that each codeword of H' and K' has even weight. H' and K' are then (8, 4) linear codes. The only possible weights for codewords in H' and K' are 0, 4, and 8; the minimum distance must therefore be 4. It is no longer the case that K' contains the codewords of H' in reverse. We then have the following generator matrices for H' and K' :

$$\underline{G}_{H'} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$$\underline{G}_{K'} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

3.3 The Golay Code

Let a and b be codewords of HI , and let x be a codeword of KI . Then the Golay Code G_{24} consists of all codewords of the form:

$$[a + x, b + x, a + b + x]$$

And a generator for G_{24} may be constructed as:

$$\underline{G}_{24} = \begin{bmatrix} \underline{G}_{H'} & \underline{0} & \underline{G}_{K'} \\ \underline{0} & \underline{G}_{H'} & \underline{G}_{K'} \\ \underline{G}_{H'} & \underline{G}_{H'} & \underline{G}_{K'} \end{bmatrix}$$

G_{24} has the following characteristics:

- It is a rate- $\frac{1}{2}$ (24, 12, 8) cyclic linear code.
- All non-zero codewords have weight divisible by 4. A code with this property may be referred to as “doubly even.”
- It can detect error-patterns with weight up to 7 and correct those with weight up to 3.
- It is self-dual – it is its own dual space (thus the generator and parity check matrices are the same).

G_{24} has many other symmetries as well. Also, if one column is removed from its generator matrix, we will have constructed G_{23} , which is a perfect code. G_{23} has minimum distance 7 and is thus also able to correct up to three errors. G_{23} and G_{24} are the only codes with their respective characteristics.

The only perfect linear binary codes are the odd-length binary repetition codes, the single error correcting Hamming codes, and the Golay code G_{23} . The ternary Golay code G_{11} is also a perfect linear code with a non-binary alphabet. There are also various nonlinear perfect codes.

4 Trellis Description of Codes

A trellis T is a directed graph containing sets of vertices (states) $V_i(T)$, distinguished by time-index i . Vertices from time-index $i - 1$ are connected to those at time-index i by labeled edges (branches) $E_i(T)$. The edge associates vertices from the two time indices with a label α drawn from a symbol alphabet which forms a group.³In the binary case, this alphabet consists of 0 and 1. This definition is illustrated in Figure 1 below.

³A group G associates a set of distinct elements with some binary operation (the group product $x \oplus y$) such that the following conditions are satisfied:

1. *Identity*: there exists an element $e \in G$ such that, for all $x \in G$, $x \oplus e = e \oplus x$.
2. *Associative Property*: for all $a, b, c \in G$, $a \oplus (b \oplus c) = (a \oplus b) \oplus c$.
3. *Inverse*: For all $a \in G$, there exists some element $a^{-1} \in G$ such that $a \oplus a^{-1} = a^{-1} \oplus a = e$.

An *Abelian* group is also commutative.

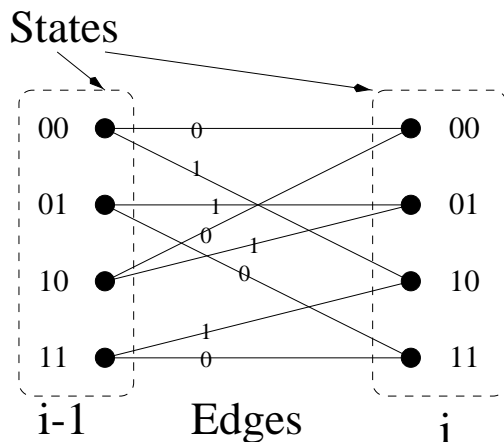


Figure 1 – Trellis Definition

A trellis may be used to describe a code C if we consider the set of connected paths through the trellis to have a one-to-one correspondence with the set of codewords in C . Branches are labeled with symbols which correspond to the signal constellation used in the coded transmission. Thus a path through the trellis corresponds to a sequence of symbols physically produced by the transmitting device.

The *state-complexity* for a time index is the number of states $|V_i(T)|$ that the trellis has at that time-index. The *state-complexity profile* is the set $\{|V_0|, |V_1|, \dots, |V_{N-1}|\}$ for a trellis with N sections.

4.1 Conventional Trellises

A conventional trellis is one in which all paths begin in a single “root” vertex, and end at a single “goal” vertex. Thus if the code has length N , then there will be N trellis sections, with a single state at $i = 0$ and another single state at $i = N$. A valid codeword thus begins in the initial zero-state, traverses a path across the branches of the trellis, and terminates in the final zero-state. A sample conventional trellis for the $(3, 1)$ binary repetition code is shown below in Figure 2:

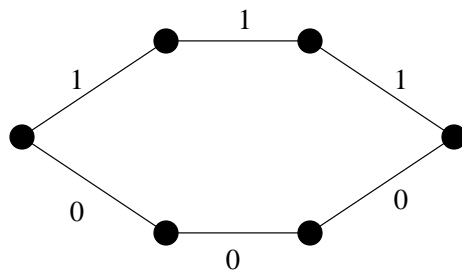


Figure 2 – Conventional Trellis

This trellis represents the code for which the only valid codewords are “111” and “000”.

4.1.1 Minimal Conventional Trellises

If x is a basis vector for a linear block code C (e.g., x is a row of C ’s generator matrix), then we define $R(x)$ as the index of the right-most non-zero coordinate in x . Similarly, $L(x)$ is the index of the left-most non-zero coordinate. We then define the *span* of x to be $Span(x) = R(x) - L(x)$.

A row of a generator matrix is considered *active* on the interval $[R(x), L(x)]$. Each column in a generator matrix represents one time-index in the trellis. If we denote by a_i the number of active rows in the trellis at time-index i , then the state-complexity for that time-index is:

$$|S_i| = 2^i$$

The *total span* of a generator matrix is then the sum of the spans of each of its rows. The complexity of a conventional trellis is related to its total span. A Minimal Span Generator Matrix (MSGM) has minimum total span and may be used to directly construct a minimal conventional trellis for the code.

A minimal span generator matrix may be produced using a simple algorithm described by Schlegel in [4, page 202]. Once an MSGM has been obtained, the minimum conventional parity-check trellis may be constructed using the method given in [4].

This construction is not used in tail-biting constructions, so the details will not be summarized. The results for the (8, 4) Hamming code are presented below. The underlined entries in the matrix represent positions in which the respective rows are active.

$$\begin{bmatrix} \underline{11} & 11 & 00 & 00 \\ \underline{01} & \underline{01} & \underline{10} & 10 \\ 00 & \underline{11} & 11 & 00 \\ 00 & 00 & \underline{11} & 11 \end{bmatrix}$$

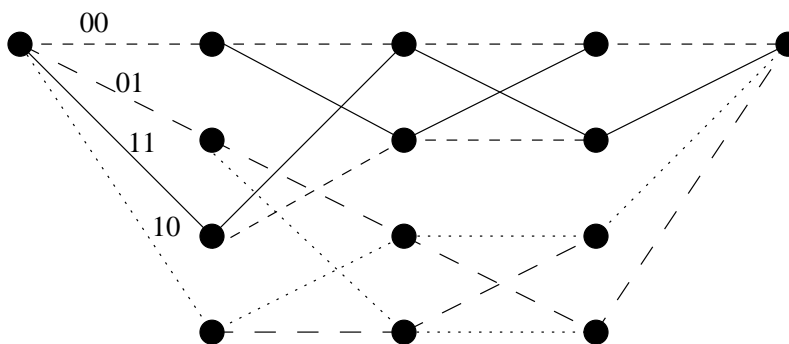


Figure 3 – Conventional Trellis for (8, 4) Hamming Code

Observe that the trellis in Figure 3 has been “contracted” – adjacent branches have been combined and labeled with their associated binary pairs. In this case, the contraction reduces the overall state-complexity of the trellis.

4.1.2 Construction by Trellis Products

There are several possible methods for constructing a trellis to represent a linear block code C based on its generator matrix. The method used by Calderbank, et al to construct the Golay code trellis [3] is construction from a set of “elementary trellises” combined through a “trellis product” operation described by Kschischang and Sorokine in [5].

Elementary trellises are first constructed to represent the different linear combinations of each individual row of the generator matrix. These elementary trellises represent subcodes of C and consequently possess a group structure. This fact may be used to combine the elementary trellises into one large trellis which completely represents C .

The elementary trellis for a given row represents all linear combinations of that row (i.e., if we let \vec{r} be the row in question, the the trellis includes \vec{r} and $\vec{r} + \vec{r}$. The elementary trellises for the (8, 4) Hamming code generator matrix (given above) are shown below:

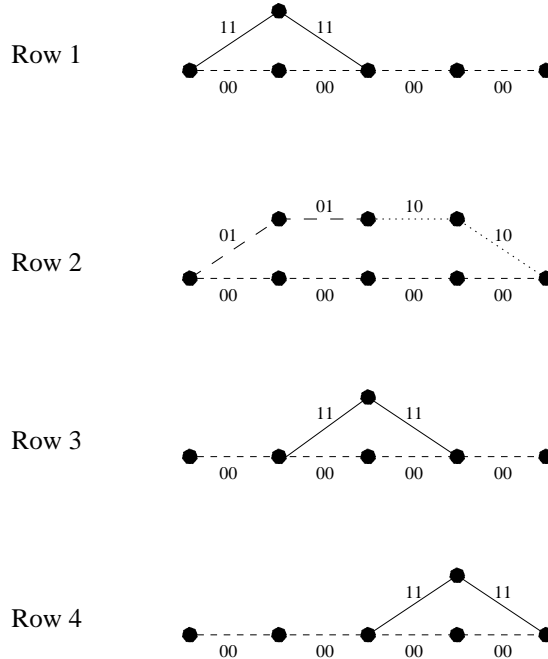


Figure 4 – Elementary Trellises for (8, 4) Hamming Code

The product of two elementary trellises $T_1 \cdot T_2$ may be taken one section at a time (the section corresponding to time-index i consists of the states at $i - 1$ together with those at i and the branches connecting them). Suppose that a branch of T_1 is represented as $\{v_1^{(i-1)}, \alpha_1, v_1^{(i)}\}$ (where α is the symbol representing the branch's label) and a branch of T_2 is represented as $\{v_2^{(i-1)}, \alpha_2, v_2^{(i)}\}$, then the product of those branches is represented in $T_1 \cdot T_2$ as $\{(v_1^{(i-1)}, v_2^{(i-1)}), \alpha_1 \oplus \alpha_2, (v_1^{(i)}, v_2^{(i)})\}$, where \oplus represents the group product associated with the branch labels.

Taking the products over all edges for all sections in the two trellises completes the product. As an example, the product of the elementary trellises for rows (1) and (2) is shown below:

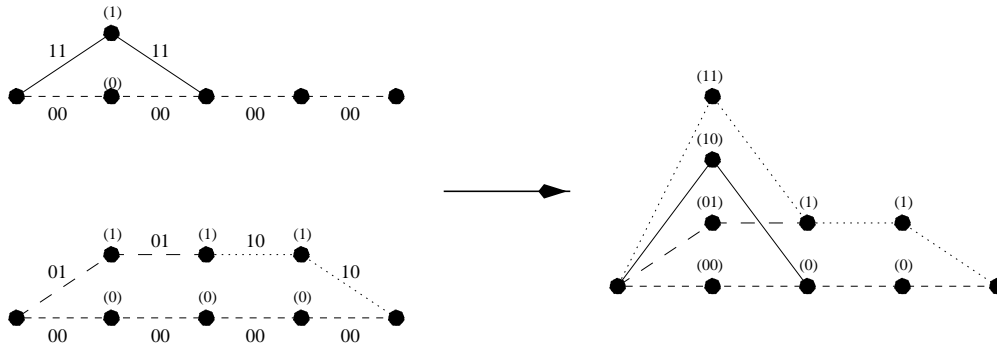


Figure 5 – Example of Trellis Product

These products may be continued on the remaining rows to produce the trellis of Figure 3 (or one equivalent to it).

4.2 Tail-Biting Trellises

A tail-biting trellis differs from a conventional trellis in that it lacks distinct root and goal vertices. The state complexity at time zero may be greater than one. The trellis is defined on a periodic time-axis, so that the final state-space equals the initial state-space.

Valid paths through a tail-biting trellis consist of those which begin and end in the same state. We may thus think of a conventional trellis as a special case of a tail-biting trellis, in which the state-complexity $|S_0| = |S_N| = 1$. Thus all valid paths are those which begin and end in the same state – that state being (0), the only state available.

To construct a tail-biting trellis from a block code’s generator matrix we must redefine the notion of a row’s span. For a conventional trellis the span is considered to be an interval $[a, b]$ in which a is always less than b . For a tail-biting trellis, though, we may regard the span as circular – it may begin at any column and “wrap around” to the beginning.

For each row, we may designate where the span begins and ends. This amounts to an arbitrary choice of the positions in which the row is active or inactive. This allows additional manipulation of state complexity and may consequently produce trellises with less total state-complexity than could a conventional trellis for the same code. These concepts will be made more clear in an example below.

4.2.1 Minimality of Tail-Biting Trellises

The MSGM no longer characterizes a minimal tail-biting trellis. Calderbank, et. al. in [3] define two measures of minimality for a tail-biting trellis. The first, μ -minimality, is defined as the minimization of the maximum state-complexity $|S_{max}|$. The second measure is π -minimality, which refers to the minimization of the product of state-complexities $\prod_{k=0}^N |S_k|$.

Minimality of conventional trellises is related to that of tail-biting trellises through the square-root bound. If $|S_{mid}|$ is the minimum state-complexity for a tail-biting trellis at its midpoint, then:

$$|S_{max}| \geq \sqrt{|S_{mid}|}$$

Where $|S_{max}|$ is the maximum state-complexity of a tail-biting trellis for the same code. It is further argued in [3] that a tail-biting trellis which meets the square-root bound with equality is both μ -minimal and π -minimal.

This bound illustrates the potential of tail-biting trellises for reducing trellis complexity, but in general the minimal construction is not known for a given code. The trellis constructed for the (24, 12, 8) binary Golay code, given in [3] and explained below, is known by this theorem to be minimal.

4.2.2 Example: Repetition Code Over F_4

The authors of [3] used the (2, 1, 2) repetition code over F_4 to illustrate the construction of tail-biting trellises from block codes using the trellis-product method. This example will be repeated here with additional detail.

F_4 is a field which possesses four elements. F_4 will be used as the basis of the hexacode and the MOG in the next section, and so will be carefully defined here.

A field is an extension of the concept of a group which adds a second operation, according to the following definition. A field F associates a set of elements with two operations: \oplus (the “additive” operation) and \odot (the “multiplicative” operation), such that:

1. The elements of F form a group under \oplus , and
2. The elements of F , excluding the additive identity element, form a group under \odot .

The field F_4 is therefore defined by the following operation tables:

\oplus	0	1	ω	$\overline{\omega}$
0	0	1	ω	$\overline{\omega}$
1	1	0	$\overline{\omega}$	ω
ω	ω	$\overline{\omega}$	0	1
$\overline{\omega}$	$\overline{\omega}$	ω	1	0

\odot	0	1	ω	$\overline{\omega}$
0	0	0	0	0
1	0	1	ω	$\overline{\omega}$
ω	0	ω	$\overline{\omega}$	1
$\overline{\omega}$	0	$\overline{\omega}$	1	ω

Now we define the $(2, 1, 2)$ repetition code over F_4 to be a code which takes a binary pair $[b_1 \ b_2]$ as its input and outputs a codeword of length two from the set $\{“00”, “11”, “\omega\omega”, “\overline{\omega}\overline{\omega}”\}$. The code is generated by the matrix:

$$\begin{bmatrix} 1 & 1 \\ \omega & \omega \end{bmatrix}$$

Conventionally, we would say that both rows are active in the first column – their spans are for each the interval $[0, 1]$. Thus for time-index 1 there are four states, and there is one state (2^0) at time-index 0 (recall that the circular time-axis for a tail-biting trellis entails that time-index 2 is the same as time-index 0).

However, since this is a tail-biting construction we may say that the span intervals of the rows are $[1, 0]$. This would make both rows active in the second column, resulting in four states at time-index 0 and one state at time-index 1. The resulting trellises for each construction are shown in Figure 6 below.

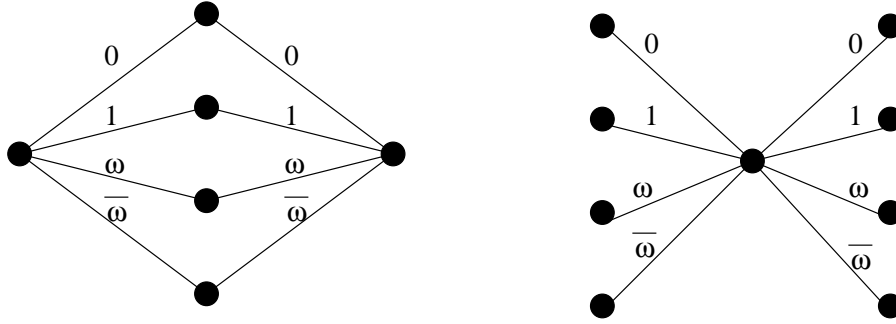


Figure 6 – Non-minimal trellises for the $(2, 1, 2)$ repetition code

The tail-biting trellis constructed here is clearly not minimal ($|S_{max}| = |S_{mid}|$). Supposing, though, that we consider the span of the first row to be $[0, 1]$ and that of the second row to be $[1, 0]$. Then the first row is active in the first column and the second row is active in the second column. The state complexity at all times would thus be 2. The resulting trellis is therefore minimal. Its elementary trellises and their product are shown below in Figure 7.

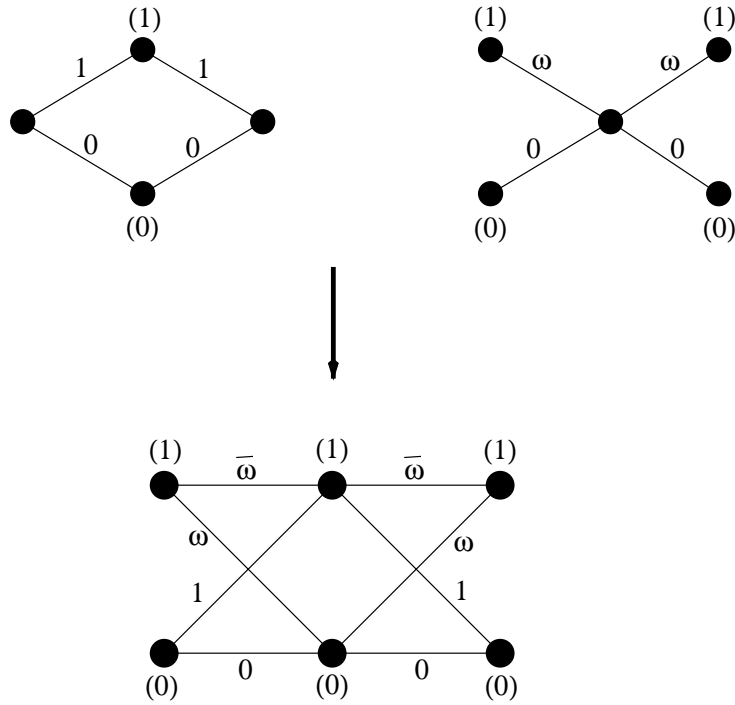


Figure 7 – Minimal Trellis for $(2, 1, 2)$ repetition code

5 The Hexacode and the MOG

F_4 is used as the symbol alphabet for the $(6, 3, 4)$ hexacode, which is used as the basis for the Miracle Octad Generator. Both the hexacode and the MOG are explored in detail in [6, chapter 11].

5.1 Hexacode

The hexacode may be defined in terms of five basic codewords and three symmetry rules. The codewords are:

- $(00\ 00\ 00)$
- $(00\ 11\ 11)$
- $(01\ 01\ \omega\varpi)$
- $(\omega\varpi\ \omega\varpi\ \omega\varpi)$
- $(11\ \omega\omega\ \varpi\varpi)$

The codewords are deliberately divided into three pairs, from which all valid hexacode words may be produced through the following transformations:

- Scalar multiplication by ω or by ϖ
- Flipping of the digits in any *two* couples
- Permutation of the couples in any way

Thus valid hexacode words are those which may be reduced through the given transformations to one of the five basic codewords.

5.2 Miracle Octad Generator and Golay Codewords

The MOG represents Golay codewords through a 6 by 4 matrix. An entry in the matrix may be chosen to represent any bit in the codeword, as long as a fixed labeling is used (rearrangement of the bits will change the valid codewords, but does not alter the properties of the code itself). Simply stated, the MOG is a method for checking a codeword to see if it is a valid Golay code word.

In the MOG, 0's and 1's are represented by blanks and non-blanks, respectively. The check is performed one column at a time. First, the number of non-blank entries in a column is written above that column. Then we sum down the column, treating the entries as elements of F_4 (from top to bottom: 0, 1, ω , ϖ), and write the result underneath the array.

This is done for each column. Finally, the number of non-blank entries in the top row is written to the right of the array. To be a Golay codeword, the array must satisfy the following conditions: the sequence on the bottom must form a valid hexacode word, and the numbers on the top, as well as that to the right, must be either all even or all odd.

As an example, we will construct the MOG for the word "110010100110000001100000" using the following labeling:

1	5	9	13	17	21
2	6	10	14	18	22
3	7	11	15	19	23
4	8	12	16	20	24

We then have:

2	2	2	0	2	0	
*	*					2
*		*		*		
	*	*		*		
1	ω	ϖ	0	ϖ	0	

Taking the bottom codeword, we may verify that it belongs to the hexacode by first multiplying by ω . This gives us " $\omega\varpi 1010$ ", which may then be permuted to give " $1010\omega\varpi$ ". Now flipping the bits of the first two couples give " $0101\omega\varpi$ ", which is one of the basic codewords. All of the numbers at the top and to the right are even, so this is a genuine Golay codeword for the specified bit ordering.

At this point it should be clear how a generator matrix for the Golay code can be produced from the MOG. The rows of the generator matrix are linearly independent codewords which form a basis for the vector subspace which contains the code. If we know the form which the matrix should take (i.e., if we know the desired spans for each row), then we could search for a bit labeling for the MOG for which each desired span interval can contain non-zero entries which support a valid codeword (entering zeros everywhere else). This is the method used in the next section.

6 Minimal Trellis for G_{24}

It is stated in [3] that the structure of the generator matrix for a minimal tail-biting Golay code trellis should have the following form:

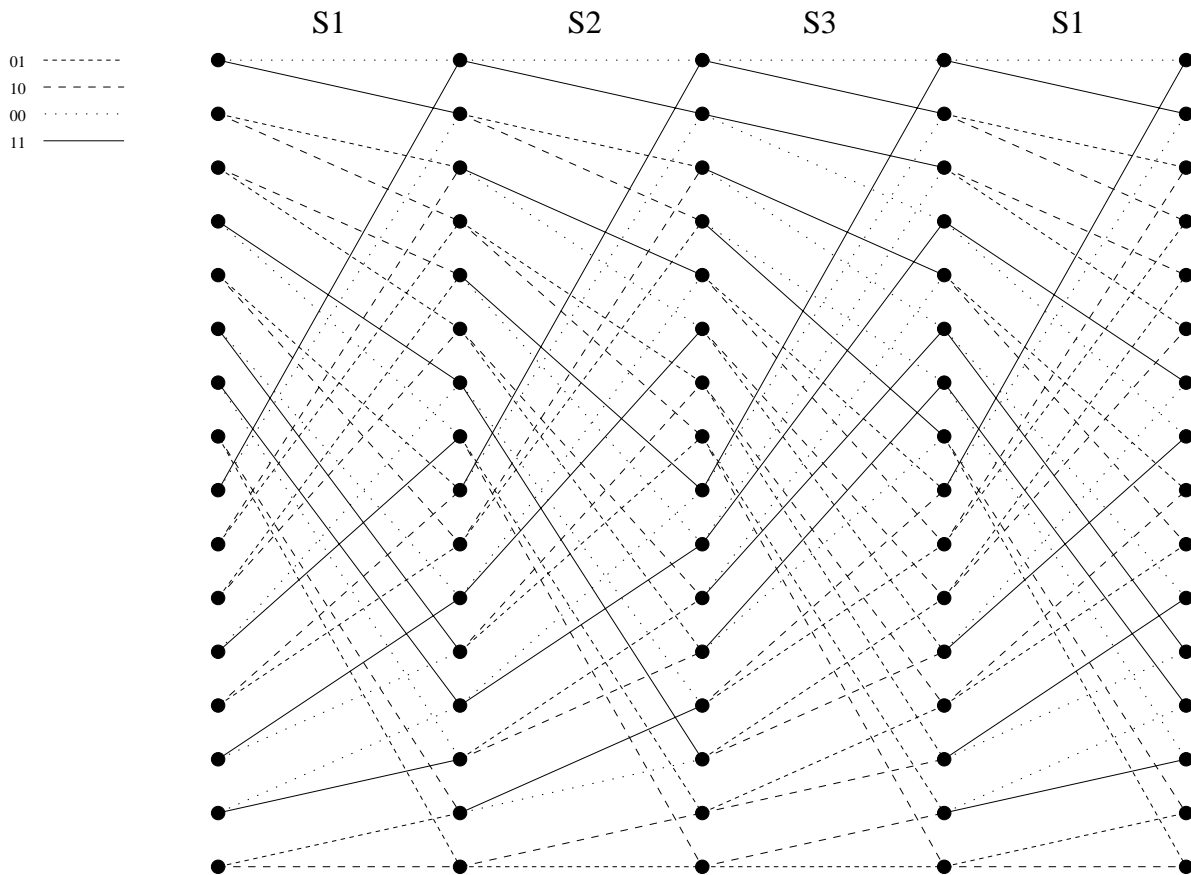


Figure 8 – Golay Code Sub-Trellis

This trellis bears a simple relationship to the source bits. Each state has two branches leaving and two arriving. Of the two leaving, the “upper” branch corresponds to a source-bit “0” in that position, and the “lower” branch corresponds to a “1”. Even numbered states receive only branches associated with “0” source bits, and states labeled with odd numbers receive only “1” source bits. Thus if we know, at each time-index, the probability distribution of the states, we need merely sum the odd-numbered state probabilities to determine the probability that the source bit at that time-index is a “1”.

References

[1] Wicker, Stephen; *Error Control Systems for Digital Communication and Storage*. Prentice-Hall, 1995.
 [2] Pretzel, Oliver; *Error-Correcting Codes and Finite Fields*. Clarendon Press, 1992.
 [3] A.R. Calderbank, G.D. Forney, Jr., and A. Vardy; “Minimal Tail-Biting Trellises: The Golay Code and More”; *IEEE Trans. on Information Theory*; vol. 45, no. 5; pp 1435-1455, July, 1999.
 [4] Schlegel, Christian; *Trellis Coding*; IEEE Press, 1997.
 [5] Frank Kschischang and Vladislav Sorokine; “On the Trellis Structure of Block Codes”, *IEEE Trans. on Information Theory*; vol. 41, no. 6; pp 1924-1937; November 1995.
 [6] J.H. Conway and N.J.A. Sloane; *Sphere Packings, Lattices and Groups*; Springer-Verlag, 1993.