Chapter 3 Codes on Graphs

In the chapter we will introduce the subject of codes on graphs. This subject essentially forms an intellectual foundation for all known classes of linear error-correcting codes, including the capacity-approaching turbo codes and low-density parity-check codes, which is the main theme of this book.

There are many styles of graphical realizations [15][23][25][86][87] of linear error-correction codes, e.g., parity-check realizations, generator realizations, and trellis (state) realizations. More generally, we consider "behavioral" realizations, in which the time axis of a trellis realization is replaced by a general unordered graph.

We will first introduce the fundamental trellis representation of linear block codes. Then, we will particularly focus on the most important aspects of Markov property and the state space theorem for linear block codes, which are the crucial basis for further investigation of the interesting topics of codes on graphs. Finally, we consider elementary linear behavioral realizations by treating the given constraints as a set of linear equations, and generalize to case where each constraint is given and well characterized by a linear code. We study in-depth on how the general linear block codes can be naturally represented by the behavioral realizations.

Since an ordinary convolutional code can be viewed as certain linear block code of infinite length, or alternatively a linear block code of finite length can be represented as a terminated convolutional code. The properties and conclusions derived from linear block

codes are also suitable for linear convolutional codes. Therefore, we only focus on the study of linear block codes in the subsequent sections.

3.1 Trellis Representation of Linear Block Codes and Decoding Complexity Profile

3.1.1 Trellis Representation of Linear Block Codes

We have known that certain binary linear block codes could be represented as terminated convolutional codes, and therefore have trellis representations as that for ordinary convolutional codes. The trellis representation for a block code is established based on its parity-check matrix. The following examples will show the trellis representation for linear block codes.

Example 3.1: Single-parity-check (SPC) (n, $n-1$, 2) code is a linear block code that adds an additional parity check bit to check n information bits. Thus, the parity-check matrix of SPC code is an n-dimensional row vector (1, 1, ….., 1), and its corresponding two-state trellis representation is shown in Fig. 3.1.

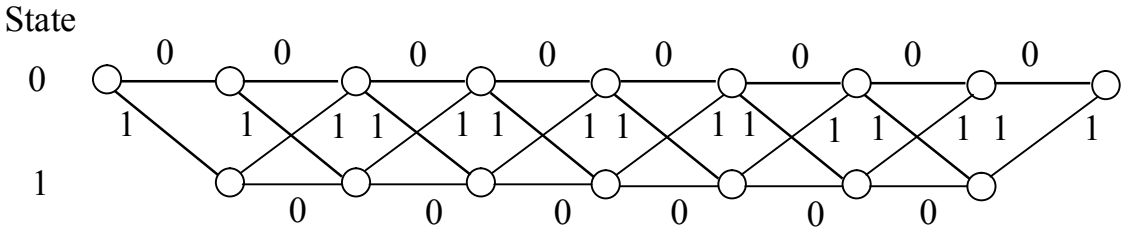


Fig. 3.1 Two-state trellis representation for a binary (n, $n-1$, 2) single-parity-check code (n=8).

We can also simplify and change the eight-section trellis representation as shown in Fig. 3-1 into a four-section trellis depicted in Fig. 3.2. Note that the resultant trellis exists parallel transmission for each branch, e.g., the output symbols are 00 and 11 for transmission between state 0 to state 0. □

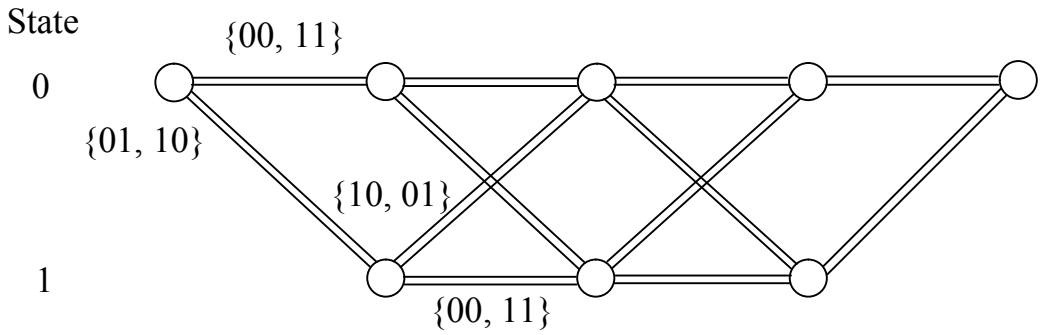


Fig. 3.2 Two-state four-section trellis representation for a binary (n, $n-1$, 2) single-parity-check code (n=7).

In general, a trellis representation of a block code is a graph like that shown in Fig. 3.1, in which there is one starting state at time 0, one ending state at time N, and state spaces S_k of size greater than one at all $n-1$ intermediate time k, $1 \le k \le n-1$. Without essential loss of generality, we always assume that the starting and ending states are zero. All edges (branches, states transitions) go from a state at some time k to a state at the next time $k+1$. Each edge is labeled by the output symbols at certain transition. The set of all possible codewords is in one-to-one correspondence with the set if all possible paths through the trellis. The codeword associated with any particular path is the sequence of corresponding output symbols at each edge.

$$H_{4\times 8}=\begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \end{pmatrix} \tag{3-1}$$

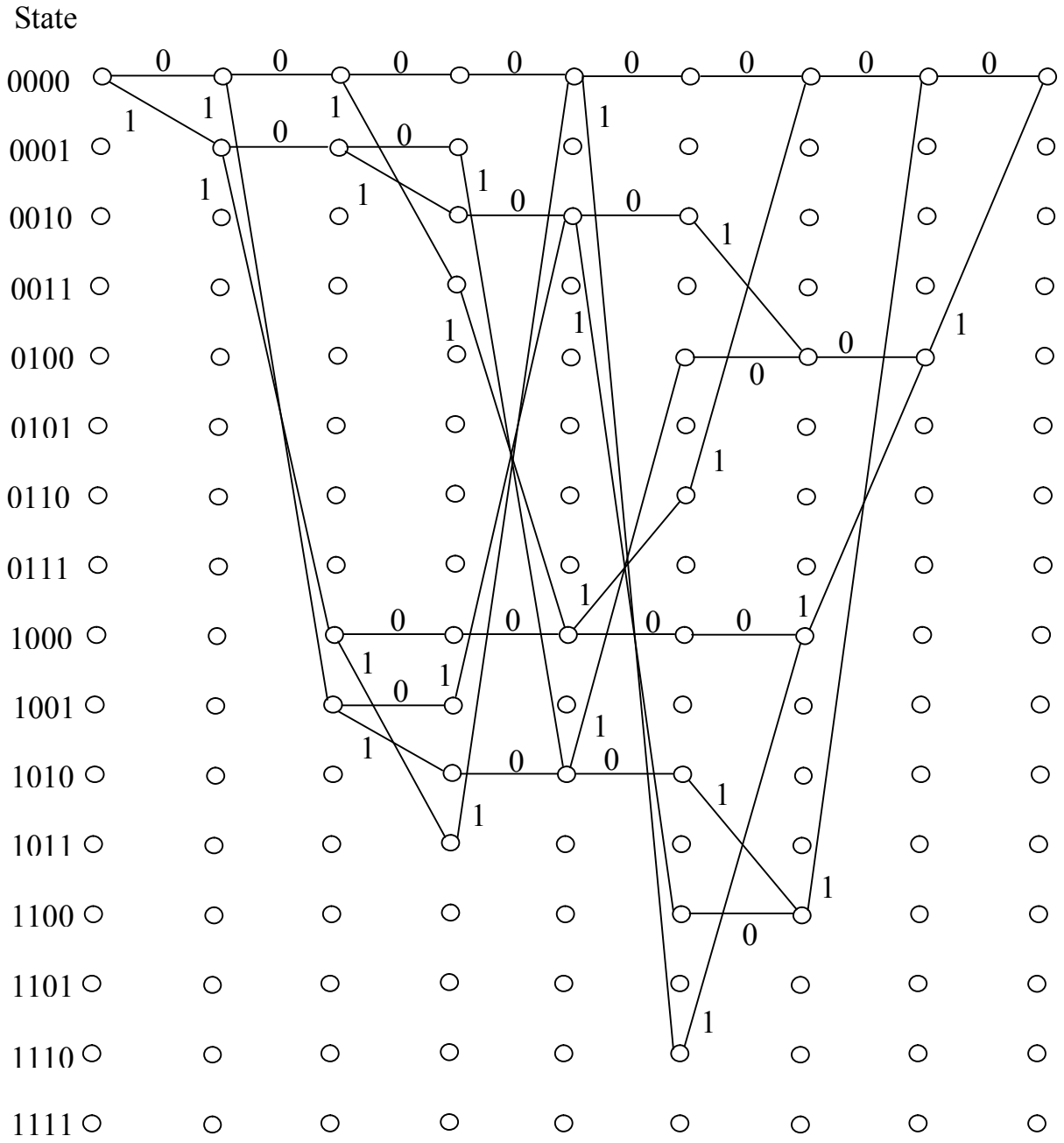
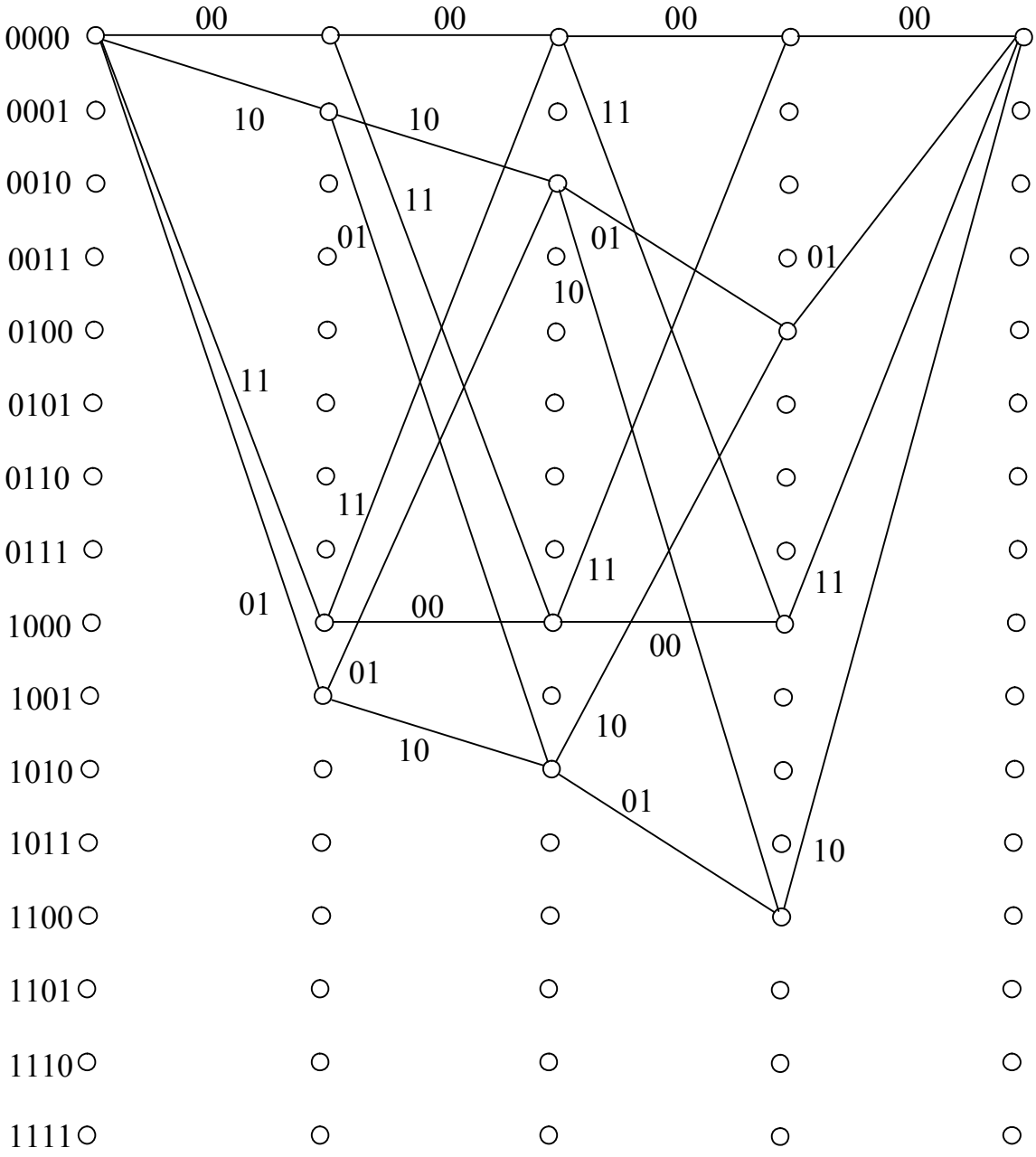
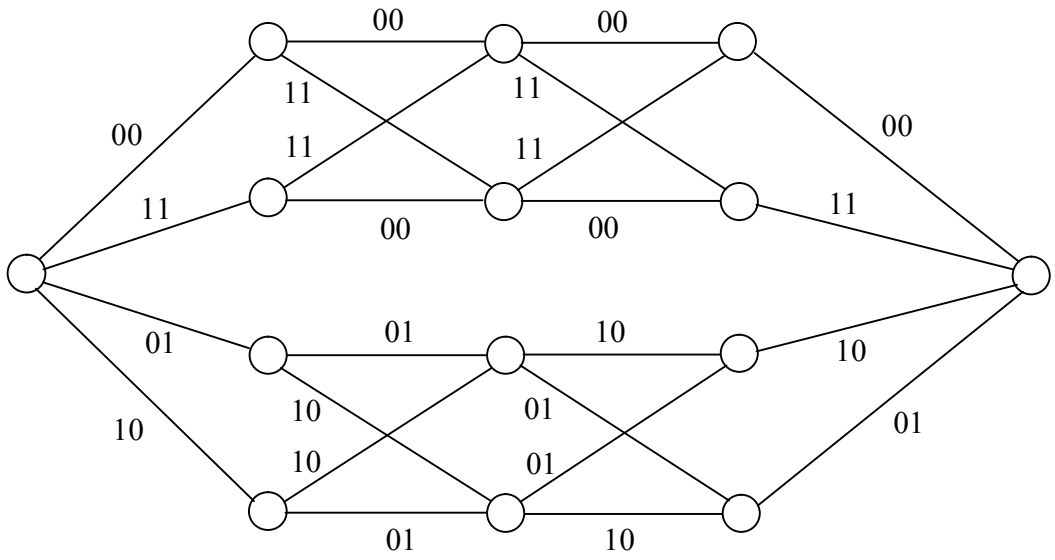


Fig. 3.3 The eight-section trellis for a binary (8, 4, 4) Reed-Muller code.

Example 3.2: Fig. 3.2 shows a 16-state trellis representation of an (8, 4, 4) binary Reed-Muller (RM) code based on the following parity-check matrix $H_{4\times 8}$ in (3-1). The 16-state 4-section trellis representation and its simplified version for the (8, 4, 4) Reed-Muller code is shown in Figs. 3.4(a) and 3.4(b).



(a)



(b)

Fig. 3.4 The four-section trellis (a) and its simplified version (b) for a binary (8, 4, 4) Reed-Muller code.

3.1.2 Decoding complexity profile

Given a trellis representation of a block code and a sequence of received symbols, the Viterbi algorithm (VA) may be used to find the trellis path with greatest metric value, i.e., to perform ML decoding of the code.

(1) State complexity profile: There are various measures of the complexity of a trellis, which are all related to VA decoding complexity. The most common is the state complexity profile, which is just the sequence of state space sizes. For instance, the state complexity profiles of the trellis representations of (8, 7, 2) SPC code are {1, 2, 2, 2, 2, 2, 2, 2, 1} and {1, 2, 2, 2, 1} for eight- and four-section shown in Figs. 3.1 and 3.2,

respectively, while the state complexity profiles are {1, 2, 4, 8, 4, 8, 4, 2, 1}, {1, 4, 4, 4, 1} for eight- and four-section (8, 4, 4) RM code shown in Figs. 3.3 and 3.4, respectively.

(2) Branch complexity profile: The branch complexity profile is the sequence of numbers of branches in each section. For example, there are {2, 4, 4, 4, 4, 4, 4, 2} and {4, 8, 8, 4} for eight- and four-section trellises of (8, 7, 2) SPC code, respectively, while there are {2, 4, 8, 8, 8, 8, 4, 2} and {4, 8, 8, 4} for eight- and four-section trellis of (8, 4, 4) RM code, respectively. The branch complexity is the maximum number of branches in any section, e.g., 4 and 8 for (8, 7, 2) SPC code and both 8 for (8, 4, 4) RM code in relation to the eight- and four-section trellises. The complexity of VA decoding is more precisely measured by the branch complexity profile, but any of these measures offers a very good idea of VA decoding complexity.

3.2 Markov Property and State Space Theorem for Linear Block Codes

3.2.1 The Partition of a Linear Block Codes by Subcodes

The state space theorem is a fundamental theorem that helps to set up a lower bound on the state space size of a trellis for a linear block code at any time, and also indicates an effective way to design a canonical minimal trellis. In this subsection we will only investigate the case for binary linear block codes. In fact, it can generalize to arbitrary linear codes and codes over groups [15].

We can think of a trellis as defining a state-space realization of a time-varying finite-state discrete-time linear system, where these terms are used as in system theory. The

time axis I of the system is some subinterval of the integers Z; i.e., a finite subinterval I= $[0, N] \subseteq Z$. In a state-space realization, a state space S_k is defined at each time $k \in$I. The Markov property of a state space is defined as follows:

Definition 3.1: The state space S_k of a system at time k has the Markov property if, given that the system is in a certain state $s_k \in S_k$ at time k, its possible future trajectories, which are the sequences of (state, input/output) alongside the trellis diagram, depend on s_k and not otherwise on the previous history of the system. In other words, the state $s_k \in S_k$ is a sufficient statistic for the past with respect to prediction of possible futures. □

Let *past* at time k be the set $P=(-\infty, k) \cap$I of time indices in I prior to time k, and the *future* at time k be set $F=[k, +\infty) \cap$I of time indices at time k or later. Given a linear code C, the state space theorem introduced in this subsection will be explicitly expressed in terms of certain linear subcodes of code C defined on the past P and future F.

(1) Shorten subcodes of code *C*: Given any sunset $J \subseteq$I, the *subcode* C_J of code C is defined as the set of codewords whose components are equal to 0 on the complementary of J in I. We may think of C_J either as a code of length $|J|$ or as a code of length |I| in which all symbols not in J equal 0. Note that it should be clear from the context whichever the two viewpoints is adopted.

The subcode C_J is evident a subset of the codewords in C that has the group property, and thus is a linear code. In coding theory, C_J is called a shorten subcode of C. The minimum Hamming distance of C_J is at least as great as that of C because $C_J \subseteq C$.

(2) Punctured subcodes of code *C*: Similarly, given $J \subseteq$ I, the projection $C_{|J}$ is defined as the set of all projections of codewords of C onto J. By projection, we force all the coordinates whose indices are not in J to be zero, or throwing away all such coordinates. Correspondingly, it is possible to think of $C_{|J}$ either as a code of length $|J|$ or as a code length |I| in which all symbols not in J equal 0. In coding theory, $C_{|J}$ is called a punctured subcode of C.

Clearly, the projection $C_{|J}$ inherits the group properties from C, and thus is also a linear code defined on J. We should bear in mind that $C_{|J}$ may not be a subcode of C even if zeros are appended after the last bit of $C_{|J}$ such that the block lengths of both codes are equal. The following example will give an explicit explanation on this fact.

Example 3.3: For the (8, 4, 4) RM code illustrated in Fig. 3.4(b), we set the "past" as the following cases:

(**1**) P={0, 1, 2, 3}: In this case the "past" means the first two time units or first four bits, then the subcode C_P forms the codewords set consisting of only two codewords {00000000, 11110000}. This code can be effectively viewed as (4, 1, 4) binary repetition code {0000, 1111} that is a shorten subcode of (8, 4, 4) RM code and defined on the past

subinterval P. The projection on this subinterval is the set $C_{|P}$={0000, 1111, 0011, 1100, 0101, 1010, 0110, 1001}, which is a (4, 3, 2) binary linear single-parity-check (SPC) code that definitely contains the (4, 1, 4) code as a subcode. Note that (4, 3, 2) binary SPC code is not a shorten subcode of (8, 4, 4) RM code.

(**2**) P={0, 1}: In this case the "past" implies the first time unit or first two bits, then the shorten subcode is an all-zero trivial code C_P ={00}, i.e., (2, 0, ∞). Note that the minimum Hamming distance of trivial linear block code with only all-zero codeword is reasonably defined as infinite. The projection on this subinterval is the set $C_{|P}$={00, 11, 01, 10} that is also a trivial binary linear code (2, 2, 1) but not a shorten subcode of (8, 4, 4) RM code. Clearly, (2, 0, ∞) code is a subcode of (2, 2, 1) code.

(**3**) P={0, 1, 2, 3, 4, 5}: In this case the "past" is the first three time units or first six bits, then the shorten subcode is C_P={000000, 111100, 001111, 110011} that is a (6, 2, 4) code. The projection on the subinterval P is the set $C_{|P}$ ={000000, 111100, 001111, 110011; 000011, 111111, 001100, 110000; 010110, 101010, 011001, 100110; 010101, 011001, 101001, 100110}, which is a (6, 4, 2) code that consists of the (6, 2, 4) code as a subcode. □

Now for any time $k\in$I, let $P$and F denote the past and future subintervals with respect to k, and let C_P , $C_{|P}$, C_F and $C_{|F}$ be the past and future subcodes and projects,

respectively. Then the linear block code C must have a generator matrix of the following form:

$$G(C)=\begin{bmatrix} G(C_P) & \mathbf{0} \\ \mathbf{0} & G(C_F) \\ G(C_{|P}/C_P) & G(C_{|F}/C_F) \end{bmatrix} \tag{3-2}$$

where $[G(C_P), \mathbf{0}]$ is a generator matrix for the past subcode C_P, $[\mathbf{0}, G(C_F)]$ is a generator matrix for the future subcode C_F, and $[G(C_{|P}/C_P), G(C_{|F}/C_F)]$ is an additional set of linearly independently generators that together with $[G(C_P), \mathbf{0}]$ and $[\mathbf{0}, G(C_F)]$ generate C. More specifically, it is clear from the generator matrix in (3-2) that $G(C_P)$ and $G(C_{|P}/C_P)$ together generate the past projection $C_{|P}$, and that $G(C_F)$ and $G(C_{|F}/C_F)$ generate the future projection $C_{|F}$.

Actually, in trellis representation C_P is the maximal subcode that starts from zero state at initial time instance and ends at zero state at k-th time instance, and C_F is the maximum subcode that starts from zero state at k-th time instance and ends at zero state at the ending time instance.

The notation $C_{|P}/C_P$ is called quotient group of C_P in $C_{|P}$ since C_P is a normal subgroup of $C_{|P}$, i.e., $C_P \lhd C_{|P}$. $G(C_{|P}/C_P)$ is a set of linearly independently generators that produce the coset leader for each coset in the quotient group $C_{|P}/C_P$. Note that based on group theory coset leader can be an arbitrary element in the coset. Similarly, $C_{|F}/C_F$ is

a quotient group of C_F in $C_{|F}$ because C_F is a normal subgroup of $C_{|F}$, i.e., $C_F \lhd C_{|F}$. $G(C_{|F}/C_F)$ is a set of linearly independently generators that produce the coset leader for each coset in the quotient group $C_{|F}/C_F$.

3.2.2 The State Codes Represented by Coset Decomposition of a Linear Block Code

The state code is a very important notation in this subsection that corresponds to the state space and also associates with the subcodes C_P and C_F.

The state code S is defined as the linear code generated by the last set of generators in (3-2), i.e., [$G(C_{|P}/C_P)$, $G(C_{|F}/C_F)$]. Clearly, the dimension of the state code is

$$\dim S = \dim C - \dim C_P - \dim C_F \tag{3-3}$$

It is evident from the trellis representation of a linear block code that we have

$$|C_{|P}/C_P| = |C_{|F}/C_F| \tag{3-4a}$$

Thus, it yields

$$\begin{aligned} rank[G(C_{|P}/C_P)] &= rank[G(C_{|F}/C_F)] \\ &= \dim S \end{aligned} \tag{3-4b}$$

This implies that the projections $S_{|P}$ and $S_{|F}$ on the past and future are linear codes with the same dimension as S, thus we get

$$\dim S = \dim S_{|P} = \dim C_{|P} - \dim C_P \tag{3-5a}$$

$$\dim S = \dim S_{|F} = \dim C_{|F} - \dim C_F \tag{3-5b}$$

Actually, a codeword of the shorten state code $S_{|P}$ is a coset leader for each coset in the quotient group $C_{|P}/C_P$, while a codeword of the shorten state code $S_{|F}$ is a coset leader for each coset in the quotient group $C_{|F}/C_F$. Although $C_{|P}/C_P$ and $C_{|F}/C_F$ can be uniquely determined if $C_{|P}$, C_P, $C_{|F}$ and C_F are completely finalized, but $G(C_{|P}/C_P)$ and $G(C_{|F}/C_F)$ may not be uniquely specified because the coset leader is not unique for each coset in $C_{|P}/C_P$ and $C_{|F}/C_F$. However, for each pair of coset leaders $s_{|P} \in S_{|P}$ for a coset in $C_{|P}/C_P$ and $s_{|F} \in S_{|F}$ for a coset in $C_{|F}/C_F$, they are one-to-one correspondence due to the entire state codeword $s \in$S produced by the generator matrix $[G(C_{|P}/C_P), G(C_{|F}/C_F)]$.

Example 3.4: We still use the (8, 4, 4) RM code and consider P in the following cases:

(**1**) P={0, 1, 2, 3}:

We have linear block codes C_P=(4, 1, 4)={0000, 1111}, $C_{|P}$=(4, 3, 2)={0000, 1111, 0011, 1100, 0101, 1010, 0110, 1001}, C_F=(4, 1, 4)={0000, 1111} and $C_{|F}$=(4, 3, 2)={0000, 1111, 0011, 1100, 0101, 1010, 0110, 1001}, respectively.

(1.a) Partition of the block code $C_{|P}$ by its subcode C_P:

The quotient group $C_{|P}/C_P$ is

$$C_{|P}/C_P=\{\Lambda_0^{(P)}, \Lambda_1^{(P)}, \Lambda_2^{(P)}, \Lambda_3^{(P)}\} \tag{3-6}$$

where the cosets $\Lambda_0^{(P)}$, $\Lambda_1^{(P)}$, $\Lambda_2^{(P)}$ and $\Lambda_3^{(P)}$ in $C_{|P}/C_P$ are

$$\Lambda_0^{(P)}=C_P=\{0000, 1111\} \tag{3-7a}$$

$$\Lambda_1^{(P)}=\{0011,\ 1100\}=\Lambda_0^{(P)}+(0011)$$

$$=\Lambda_0^{(P)}+(1100) \tag{3-7b}$$

$$\Lambda_2^{(P)}=\{0101,\ 1010\}=\Lambda_0^{(P)}+(0101)$$

$$=\Lambda_0^{(P)}+(1010) \tag{3-7c}$$

$$\Lambda_3^{(P)}=\{0110,\ 1001\}=\Lambda_0^{(P)}+(0110)$$

$$=\Lambda_0^{(P)}+(1001) \tag{3-7d}$$

It is clear from Fig. 3.4(b) that all the codewords of $C_{|P}$ in the coset $\Lambda_0^{(P)}$ end at the 1st state at the 2nd time instance, while the codewords of $C_{|P}$, which belong to the cosets $\Lambda_1^{(P)}$, $\Lambda_2^{(P)}$ and $\Lambda_3^{(P)}$, respectively, end at the 2nd, 3rd, 4th state, correspondingly, at the 2nd time instance. All the codewords of $C_{|P}$ initially start from zero state.

(1.b) Partition of the linear block code $C_{|F}$ by its subcode C_F:

Similarly, the quotient group $C_{|F}/C_F$ is

$$C_{|F}/C_F=\{\Lambda_0^{(F)},\ \Lambda_1^{(F)},\ \Lambda_2^{(F)},\ \Lambda_3^{(F)}\} \tag{3-8}$$

where the cosets $\Lambda_0^{(F)}$, $\Lambda_1^{(F)}$, $\Lambda_2^{(F)}$ and $\Lambda_3^{(F)}$ in $C_{|F}/C_F$ are

$$\Lambda_0^{(F)}=C_F=\{0000,\ 1111\} \tag{3-9a}$$

$$\Lambda_1^{(F)}=\{0011,\ 1100\}=\Lambda_0^{(F)}+(0011)$$

$$=\Lambda_0^{(F)}+(1100) \tag{3-9b}$$

$$\Lambda_2^{(F)}=\{0101,\ 1010\}=\Lambda_0^{(F)}+(0101)$$

$$= \Lambda_0^{(F)} + (1010) \tag{3-9c}$$

$$\Lambda_3^{(F)} = \{0110,\ 1001\} = \Lambda_0^{(F)} + (0110)$$

$$= \Lambda_0^{(F)} + (1001) \tag{3-9d}$$

Evidently, as indicated by Fig. 3.4(b) all the codewords of $C_{|F}$ in the coset $\Lambda_0^{(F)}$ start from the 1st state at the 2nd time instance, while the codewords of $C_{|F}$, which belong to the cosets $\Lambda_1^{(F)}$, $\Lambda_2^{(F)}$ and $\Lambda_3^{(F)}$, respectively, start from the 2nd, 3rd, 4th state, correspondingly, at the 2nd time instance. All the codewords of $C_{|F}$ finally end at zero state.

(1.c) The relationship between $C_{|P}/C_P$ and $C_{|F}/C_F$:

Let the coset $\Lambda_i^{(P)}$ (i=0, 1, 2, 3) in $C_{|P}/C_P$ be in one-to-one correspondence with the coset $\Lambda_i^{(F)}$ in $C_{|F}/C_F$, we may find that the state, which all the codewords in $\Lambda_i^{(P)}$ end at the 2nd time instance, is the one that all the codewords in $\Lambda_i^{(F)}$ start at the 2nd time instance.

Based on (3-7) and (3-9) we may express the entire (8, 4, 4) RM code by a simplified two-section trellis representation shown in Fig. 3.5, where the coset leaders for the quotient groups $C_{|P}/C_P$ and $C_{|F}/C_F$ are the same, i.e., (0000), (1100), (1010) and (0110). By the above trellis representation in Fig. 3.5 the generator matrix of the (8, 4, 4) RM code can be expressed as

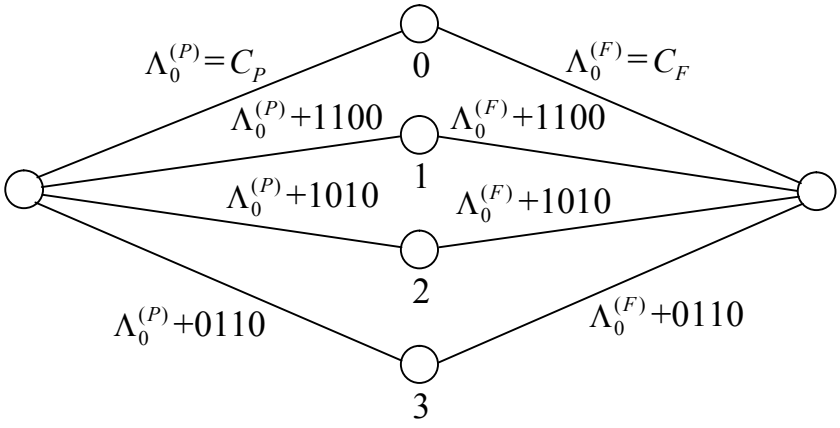


Fig. 3.5 The two-section trellis representation for (8, 4, 4) RM code by means of subcode and cosets.

$$G=\begin{bmatrix}1&1&1&1&0&0&0&0\\0&0&0&0&1&1&1&1\\1&0&1&0&1&0&1&0\\0&1&1&0&0&1&1&0\end{bmatrix} \tag{3-10}$$

where the component matrices in the form (3-2) are

$$G(C_P)=G(C_F)=\begin{bmatrix}1&1&1&1\end{bmatrix} \tag{3-11a}$$

$$G(C_{|P}/C_P)=G(C_{|F}/C_F)=\begin{bmatrix}1&0&1&0\\0&1&0&1\end{bmatrix} \tag{3-11b}$$

Meanwhile the generator matrix of the (8, 4, 4) RM code can be also expressed as

$$G=\begin{bmatrix}1&1&1&1&0&0&0&0\\0&0&0&0&1&1&1&1\\1&1&0&0&1&1&0&0\\0&1&1&0&0&1&1&0\end{bmatrix} \tag{3-12}$$

where the component matrices in the form (3-2) are

$$G(C_P)=G(C_F)=\begin{bmatrix}1 & 1 & 1 & 1\end{bmatrix} \tag{3-13a}$$

$$G(C_{|P}/C_P)=G(C_{|F}/C_F)=\begin{bmatrix}1 & 1 & 0 & 0\\ 0 & 1 & 1 & 0\end{bmatrix} \tag{3-13b}$$

Both (3-11b) and (3-12b) enable to generate the specified coset leaders (0000), (1100), (1010), (0110) for the cosets in the quotient groups $C_{|P}/C_P$ and $C_{|F}/C_F$, and the coset leader for $C_{|P}/C_P$ is associated with the corresponding one for $C_{|F}/C_F$ by the generator submatrix $[G(C_{|P}/C_P), G(C_{|F}/C_F)]$.

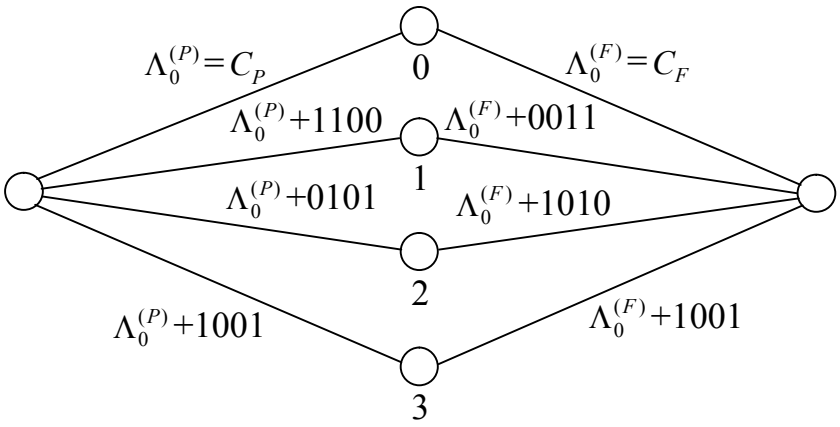


Fig. 3.6 The two-section trellis representation for (8, 4, 4) RM code by means of subcode and cosets.

(1.d) The different version of simplified two-section trellis and generator matrix:

Alternatively, the (8, 4, 4) RM code can be expressed by another simplified two-section trellis representation shown in Fig. 3.6, where the coset leaders (0000), (1100), (0101)

and (1001) for $C_{|P}/C_P$ and the coset leaders (0000), (0011), (1010) and (1001) for $C_{|F}/C_F$.

Based on Fig. 3.6 the generator matrix of the (8, 4, 4) RM code is expressed as

$$G=\begin{bmatrix}1&1&1&1&0&0&0&0\\0&0&0&0&1&1&1&1\\1&1&0&0&0&0&1&1\\0&1&0&1&1&0&1&0\end{bmatrix} \tag{3-14}$$

where the component matrices in the form (3-2) are

$$G(C_P)=G(C_F)=\begin{bmatrix}1&1&1&1\end{bmatrix} \tag{3-15a}$$

$$G(C_{|P}/C_P)=\begin{bmatrix}1&1&0&0\\0&1&0&1\end{bmatrix} \tag{3-15b}$$

$$G(C_{|F}/C_F)=\begin{bmatrix}0&0&1&1\\1&0&1&0\end{bmatrix} \tag{3-15c}$$

(**2**) $P=\{0, 1\}$:

We have linear block codes $C_P=(2, 0, \infty)=\{00\}$, $C_{|P}=(2, 2, 1)=\{00, 11, 01, 10\}$, $C_F=(6, 2, 4)=\{000000, 111100, 001111, 110011\}$ and $C_{|F}=(6, 4, 2)=\{000000, 111100, 001111, 110011; 110000, 001100, 111111, 000011; 011010, 100110, 010101, 101001; 101010, 010110, 100101, 011001\}$, respectively.

The quotient group $C_{|P}/C_P$ is

$$C_{|P}/C_P=\{\Lambda_0^{(P)}, \Lambda_1^{(P)}, \Lambda_2^{(P)}, \Lambda_3^{(P)}\} \tag{3-16}$$

where the cosets $\Lambda_0^{(P)}$, $\Lambda_1^{(P)}$, $\Lambda_2^{(P)}$ and $\Lambda_3^{(P)}$ in $C_{|P}/C_P$ are

$$\Lambda_0^{(P)} = C_P = \{00\} \tag{3-17a}$$

$$\Lambda_1^{(P)} = \{11\} = \Lambda_0^{(P)} + (11) \tag{3-17b}$$

$$\Lambda_2^{(P)} = \{01\} = \Lambda_0^{(P)} + (01) \tag{3-17c}$$

$$\Lambda_3^{(P)} = \{10\} = \Lambda_0^{(P)} + (10) \tag{3-17d}$$

Similarly, the quotient group $C_{|F}/C_F$ is

$$C_{|F}/C_F = \{\Lambda_0^{(F)},\ \Lambda_1^{(F)},\ \Lambda_2^{(F)},\ \Lambda_3^{(F)}\} \tag{3-18}$$

where the cosets $\Lambda_0^{(F)}$, $\Lambda_1^{(F)}$, $\Lambda_2^{(F)}$ and $\Lambda_3^{(F)}$ in $C_{|F}/C_F$ are

$$\Lambda_0^{(F)} = C_F = \{000000,\ 111100,\ 001111,\ 110011\} \tag{3-19a}$$

$$\begin{aligned}\Lambda_1^{(F)} &= \{110000,\ 001100,\ 111111,\ 000011\} \\ &= \Lambda_0^{(F)} + (110000) = \Lambda_0^{(F)} + (001100) \\ &= \Lambda_0^{(F)} + (111111) = \Lambda_0^{(F)} + (000011)\end{aligned} \tag{3-19b}$$

$$\begin{aligned}\Lambda_2^{(F)} &= \{011010,\ 100110,\ 010101,\ 101001\} \\ &= \Lambda_0^{(F)} + (011010) = \Lambda_0^{(F)} + (100110) \\ &= \Lambda_0^{(F)} + (010101) = \Lambda_0^{(F)} + (101001)\end{aligned} \tag{3-19c}$$

$$\begin{aligned}\Lambda_3^{(F)} &= \{101010,\ 010110,\ 100101,\ 011001\} \\ &= \Lambda_0^{(F)} + (101010) = \Lambda_0^{(F)} + (010110) \\ &= \Lambda_0^{(F)} + (100101) = \Lambda_0^{(F)} + (011001)\end{aligned} \tag{3-19d}$$

Based on the same principle, we can demonstrate the (8, 4, 4) RM code by a simplified two-section trellis representation shown in Fig. 3.7, where the coset leaders (00), (11), (01) and (10) for $C_{|P}/C_P$ and the coset leaders (000000), (110000), (011010) and (101010) for $C_{|F}/C_F$.

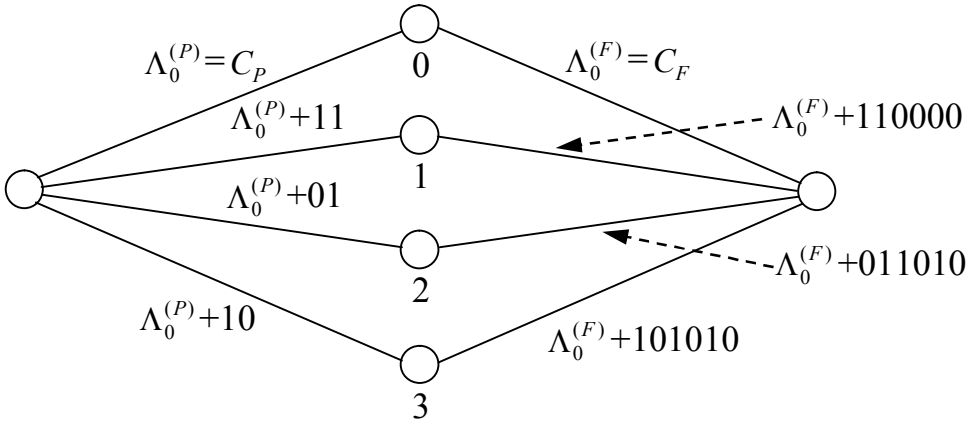


Fig. 3.7 The two-section trellis representation for (8, 4, 4) RM code by means of subcode and cosets.

The generator matrix of the (8, 4, 4) RM code is given

$$G=\begin{bmatrix}0&0&1&1&1&1&0&0\\0&0&0&0&1&1&1&1\\1&1&1&1&0&0&0&0\\0&1&0&1&1&0&1&0\end{bmatrix} \tag{3-20}$$

where the component matrices in the form (3-2) are

$$G(C_F)=\begin{bmatrix}1&1&1&1&0&0\\0&0&1&1&1&1\end{bmatrix} \tag{3-21a}$$

$$G(C_{|P}/C_P)=\begin{bmatrix}1&1\\0&1\end{bmatrix} \tag{3-21b}$$

$$G(C_{|F}/C_F)=\begin{bmatrix}1 & 1 & 0 & 0 & 0 & 0\\ 0 & 1 & 1 & 0 & 1 & 0\end{bmatrix} \tag{3-21c}$$

Note that for any realization by the generator matrix, formulated by (3-10), (3-12), (3-14) and (3-20), the state code S is always generated by the last two generators and has dimension 2. The past and future projections of the state code $S_{|P}$ and $S_{|F}$ evidently also have dimension 2. The difference among them only lies in the dimension of subcodes C_P and C_F, e.g., we have $\dim C_{|P}=2/\dim C_P=0$ and $\dim C_{|F}=4/\dim C_F=2$ for P={0, 1} and realization by Fig. 3.7, in case of P={0, 1, 2, 3} and realizations by Figs. 3.5 and 3.6 we all have $\dim C_{|P}=\dim C_{|F}=3$ and $\dim C_P=\dim C_F=1$. □

In fact, the state code **S** is constructed by a union of Cartesian products of the related coset leaders of cosets in $C_{|P}/C_P$ and $C_{|F}/C_F$, which form a linear group code under the modulo-2 addition. After the introduction of the intuitive example as above we may conclude the Markov property by the following theorem [15], which can be easily proved by the essential group theory.

Lemma 3.1 (Markov property): For all the codewords $\mathbf{c}\in C$ that are associated with a given state codeword $\mathbf{s}\in S$, the past projection $\mathbf{c}_{|P}\in C_{|P}$ has the same set of possible future trajectories. On the other hand, of two codewords $\mathbf{c}$ and $\mathbf{c}'$ are associated with different state codes, then the sets of possible future trajectories of $\mathbf{c}_{|P}$ and $\mathbf{c}'_{|P}$ are disjoint.

Proof: If a codeword $\mathbf{c} \in C$ is associated with a codeword $\mathbf{s} \in S$, then based on the generator matrix given by (3-2), we have

$$\mathbf{c} = \mathbf{c}_P + \mathbf{c}_F + \mathbf{s} \tag{3-22}$$

for some $\mathbf{c}_P \in C_P$ and $\mathbf{c}_F \in C_F$. Hence,

$$\mathbf{c}_{|P} = \mathbf{c}_P + \mathbf{s}_{|P} \tag{3-23a}$$

$$\mathbf{c}_{|F} = \mathbf{c}_F + \mathbf{s}_{|F} \tag{3-23b}$$

Thus, for every such $\mathbf{c}_{|P}$ that is associated with the codeword $\mathbf{s} \in S$, which implies that both $\mathbf{s}_{|P}$ and $\mathbf{s}_{|F}$ are finalized based on the generator matrix given by (3-2). The set of possible future trajectories is the same, namely the coset $C_F + \mathbf{s}_{|F} = \{\mathbf{c}_F + \mathbf{s}_{|F} \,|\, \mathbf{c}_F \in C_F\}$.

If $\mathbf{c}$ and $\mathbf{c}'$ are associated with different state codewords $\mathbf{s}$ and $\mathbf{s}'$, then the sets of possible future trajectories are $C_F + \mathbf{s}_{|F}$ and $C_F + \mathbf{s}'_{|F}$, respectively.

Next, we will prove that the difference $\mathbf{s}_{|F} - \mathbf{s}'_{|F}$ is not a codeword in C_F. By the generator matrix in (3-2), the relevant information blocks with respect to $\mathbf{s}$ and $\mathbf{s}'$ are different due to $\mathbf{s} \neq \mathbf{s}'$. If $\mathbf{s}_{|F} - \mathbf{s}'_{|F} \in C_F$, then the generators in $G(C_F)$ and $G(C_{|F} / C_F)$ are linearly dependent, which is contradictory with the fundamental definition of a generator matrix for linear block code. Therefore, we conclude that $\mathbf{s}_{|F} - \mathbf{s}'_{|F} \notin C_F$, which means that the cosets $C_F + \mathbf{s}_{|F}$ and $C_F + \mathbf{s}'_{|F}$ are disjoint. □

Moreover, we can further rewrite (3-22) as follows

$$\mathbf{c}=\mathbf{c}_P+\mathbf{c}_F+\mathbf{s}=(\mathbf{c}_P+\mathbf{s}_{|P})+(\mathbf{c}_F+\mathbf{s}_{|F})$$

$$=\mathbf{c}_{|P}+\mathbf{c}_{|F} \tag{3-24}$$

Thus, a codeword $\mathbf{c}\in C$ can be expressed by (3-24) if and only if $\mathbf{c}_{|P}\in C_{|P}$ is in the coset $C_P+\mathbf{s}_{|P}$ and $\mathbf{c}_{|F}\in C_{|F}$ is in the coset $C_F+\mathbf{s}_{|F}$.

Generically, the set of past projections of the codewords associated with a given state codeword $\mathbf{s}\in\mathbf{S}$ is the coset $C_P+\mathbf{s}_{|P}=\{\mathbf{c}_P+\mathbf{s}_{|P}\,|\,\mathbf{c}_P\in C_P\}$. For any past projection in this set, we have the same set of possible future trajectories, i.e., coset $C_F+\mathbf{s}_{|F}$. Moreover, these past and future subsets are disjoint. Therefore, the entire code may be written as a disjoint union of Cartesian products of past and future subsets

$$C=\bigcup_{\mathbf{s}\in S}(C_P+\mathbf{s}_{|P})\times(C_F+\mathbf{s}_{|F}) \tag{3-25}$$

Fig. 3.8 depicts a generic two-section trellis that illustrates this Cartesian-product decomposition for a general linear block code. The states are labelled by the state codewords $\mathbf{s}\in\mathbf{S}$ to which they correspond. Each edge represents a coset of parallel projections of codewords, i.e., $C_P+\mathbf{s}_{|P}$ for past edges and $C_F+\mathbf{s}_{|F}$ for future edges. The particular path corresponding to $\mathbf{s}=\mathbf{0}$ is shown to represent the subcode C_P+C_F, while a generic path corresponding to a general state $\mathbf{s}$ stands for the subset of codewords $C_P+C_F+\mathbf{s}$.

It should now be clear that the code C has a trellis representation with a state space S_k at time k that is in one-to-one correspondence with the state code S., and it has no trellis representation with fewer states at time k due to the fact that

$$|S|=|C_{|P}/C_P|=|C_{|F}/C_F| \tag{3-26}$$

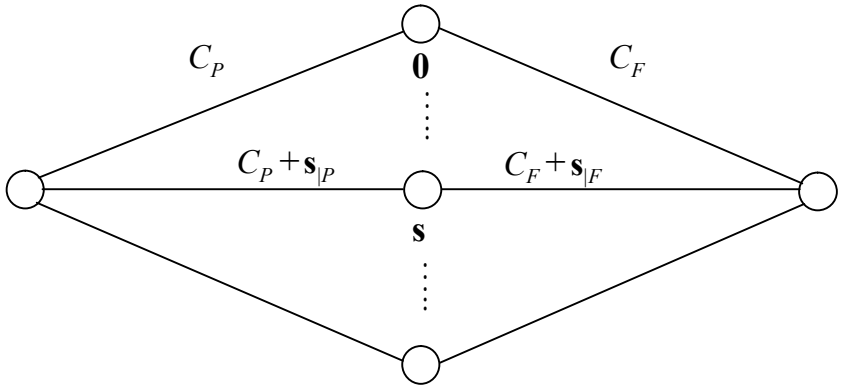


Fig. 3.8 A generic tw0-section trellis representation for a general linear block code.

Finally, we summarize this development in the following theorem:

Theorem 3.1 (State space theorem for binary linear block codes) Let C be a binary linear block code defined on a time axis I=[0, N], let $k\in$I, and let C_P, $C_{|P}$, C_F, $C_{|F}$ and S be the past and future subcodes, projections and state code relative to time k, respectively. Then there exists a trellis representation for C with $|S_k|$ states at time k, and no trellis representation with fewer states, where $\dim S_k$ and $|S|$ is given by the following expressions:

$$\dim S=\dim C-\dim C_P-\dim C_F$$

$$= \dim C_{|P} - \dim C_P$$

$$= \dim C_{|F} - \dim C_F \tag{3-27}$$

and

$$|S| = S_k = 2^{\dim S}$$

$$= |C_{|P} / C_P| = |C_{|F} / C_F| \tag{3-28}$$

3.2.3 Trellis-Oriented Generator Matrices and Branch Space

In this section we will first introduce a trellis-oriented generator matrix that can present the parameters of trellis representation of a linear block code directly, such as the dimensions of subcode and state code, etc. Then we will introduce the branch space that is deeply associated with branch complexity mentioned in Section 3.1.2 as an effective measure of Viterbi algorithm decoding complexity.

(1) Trellis-oriented generator matrix

In order to present the trellis-oriented generator matrix, the two fundamental definitions are first given as follows

Definition 3.2: The span of a codeword is defined as the interval from its first to last nonzero symbols. Its effective length is defined as the length of this span.

Definition 3.3: The trellis-oriented (or minimum-span) generator is defined as a set of $k = \dim C$ linearly independent generators whose effective lengths are as short as possible.

A trellis-oriented generator matrix may be found by first finding all codewords with effective length 1, then all codewords of effective length 2 that are not linearly dependent

on codewords of effective length 1, ..., all codewords of effective length i that are not linearly dependent on codewords of lower effective length, …., until we have k independent generators.

The following theorem shows how to check whether a given generator matrix is trellis-oriented, and also suggests how to reduce any given generator matrix to one that is trellis-oriented.

Theorem 3.2 A set of k linearly independent generators is a trellis-oriented generator matrix if and only if the starting times of all spans are distinct and the ending times of all spans are distinct.

Proof: If all starting times and ending times are distinct, then given a linear combination (with nonzero coefficients) of a certain subset of generators, the starting time and ending time of the combination are the least and greatest starting and ending times of the given subset of generators. It follows that the generators that combine to form any non-generator codeword have effective lengths no greater than the effective length of that codeword, so the given generators are indeed a set of generators whose effective lengths are as short as possible, which can lead to a trellis-oriented generator matrix.

Conversely if two starting or ending times are not distinct, then the sum of the two corresponding generators is a codeword whose effective length is shorter than that of at least one of the generators. If this generator is replaced by this codeword, then we may obtain a set of linearly independent generators of which one has a shorter effective length,

so the original set is not trellis-oriented. Therefore, the generators in a trellis-oriented generator matrix must have distinct starting and ending times of all spans. □

The important property of a trellis-oriented generator matrix used in the first part of the proof is that the starting and ending times of a linear combination (with nonzero coefficients) of a subset of generators are the earliest starting time and latest ending time of the component generators, respectively.

The second part of the proof suggests a simple greedy algorithm for finding a trellis-oriented generator matrix from a given generator matrix. If the starting or ending times of two generators in the given matrix are not distinct, then replace the generator with greater effective length by the sum of the two generators, which must necessarily have a shorter effective length. This algorithm must terminate after a finite number of steps in a trellis-oriented generator matrix. Based on above, we state the following theorem:

Theorem 3.3 Given a trellis-oriented generator matrix for a linear block C, if $[k, k'] \subseteq \mathrm{I}$ is any subinterval of the time axis I, then the subcode $C_{[k,k']}$ is the set of all linear combinations of generators whose spans are contained in $[k, k']$.

According to the above theorem it is easily to obtain the dimensions of each past and future subcode directly from a trellis-oriented generator matrix. Moreover, for any partition into past and future, the state code S is generated by those generators that lie neither wholly in the past nor wholly in the future. The dimension of the minimum state space is the number of such active generators.

Example 3.5 For the (8, 7, 2) single-parity-check (SPC) code shown in Fig. 3.1, the trellis-oriented generator matrix is

$$G=\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \tag{3-29}$$

Evidently, each row vector in (3-29) is a codeword of (8, 7, 2) SPC code, whose codeword bits within its span is depicted along the dashed line shown in Fig. 3.9. For example, the codeword (11000000) is associated with the directed line in the figure.

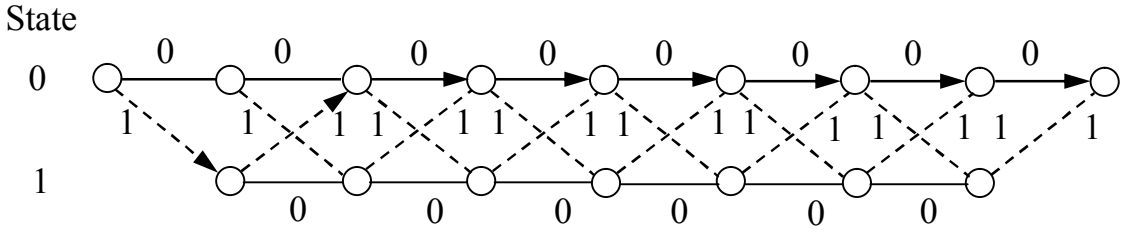


Fig. 3.9 The codewords from the row vectors of the trellis-oriented generator matrix and the codeword bits within the spans depicted in the eight-section trellis representation of the (8, 7, 2) SPC code.

At each time instance k, corresponding to a nontrivial state space, only one generator is active, which lies neither wholly in the past nor wholly in the future, thus the state space S_k has dimension 1. For instance, for k=4 (or P={0, 1, 2, 3}) the only active generator is

0001100, i.e., the fourth generator in the trellis-oriented generator matrix. The dimension of subcode C_P is 3 since the spans of the first three generators are contained in P.

Example 3.6 The standard generator matrix for the (8, 4, 4) RM code is

$$G=\begin{bmatrix}1&1&1&1&0&0&0&0\\1&0&1&0&1&0&1&0\\1&1&0&0&1&1&0&0\\1&1&1&1&1&1&1&1\end{bmatrix} \tag{3-30}$$

The ending times are all distinct, but the starting times are all the same. Adding the first generator to all other results in

$$G=\begin{bmatrix}1&1&1&1&0&0&0&0\\0&1&0&1&1&0&1&0\\0&0&1&1&1&1&0&0\\0&0&0&0&1&1&1&1\end{bmatrix} \tag{3-31}$$

All starting and ending times are now distinct, so this generator matrix is trellis-oriented. There are two active generators in (3-31) at each time k associated with a nontrivial state space in the four-section trellis representation. So the state space S_k has dimension 2. For example, the first two generators are active for k=2 (P={0, 1}), which leads to $\dim S_2$=2. The dimension of subcode C_P is 0 because the generator whose span is within P does not exist in (3-31). On the other hand, there also has two active generators for k=4 (P={0, 1, 2, 3}), i.e., the second and third generator in (3-31), which also results in $\dim S_4$=2. The dimension of subcode C_P is 1 since only the first generator whose span is within P exists in this trellis-oriented generator matrix.

(2) **Branch space and its generalized form**

Most of the work on trellis complexity has focused on state complexity, however, branch complexity is in some respects more fundamental for measure of Viterbi algorithm decoding complexity. Thus, we will present an insight view on the branch space in the subsequent part.

The time axis for branch is not the same as the time axis I=[0, *N*] for state. Branches occur at symbol times, while states occur between symbol times. Thus, there are only *N* branch times, say [0, N), while there are *N*+1 state time.

A branch at time *k* nay be identified by a triple (s_k, c_k, s_{k+1}), where (s_k, s_{k+1}) is a valid state transition, and c_k is a valid code symbol that may be generated during that transition. There may be more than one branch (parallel transition) associated with a given state transition, if there is more than one possible output during that transition. The branch space at time instance *k* is the set of all possible branches, $B_{k,1}=\{(s_k, c_k, s_{k+1})\}\subseteq S_k\times C_k\times S_{k+1}$.

In order to well understand the branch space, we will first consider a generalized linear space defined by

$$B_{k,j}=\{(s_k, c_k, s_{k+1}, c_{k+1}, \ldots.., s_{k+j-1}, c_{k+j-1}, s_{k+j})\}\subseteq\left(\prod_{i=k}^{k+j-1}(S_i\times C_i)\right)\times S_{k+j} \qquad (3\text{-}32)$$

where the set $B_{k,j}$ forms all the combinations (trajectories) of symbol and state variables through the trellis subject to starting from time *k* and ending at *k*+*j* with *j* symbol

durations. Evidently, the branch space $B_{k,1}$ is only a special case of $B_{k,j}$ with one symbol duration, i.e., j=1. Usually, we simply denote $B_{0,N}$ and $B_{k,1}$ as B and B_k, respectively, the latter one is also called constraint code in the Subsection 3.3.2.

According to the Lemma 3.1 any codeword $\mathbf{c}$ of linear block code C of block length N can be expressed as

$$\mathbf{c}=\mathbf{c}_{[0,k)}+\mathbf{s}_{0,k}+\mathbf{c}_{[k,N)} \tag{3-33}$$

where $\mathbf{c}_{[0,k)} \in C_{[0,k)}$ and $\mathbf{c}_{[k,N)} \in C_{[k,N)}$, the subscripts indicate the subsets of the time indices I, at which the subcodes are defined, the subscript of the codeword of state code $\mathbf{s}_{0,k} \in S_{0,k}$ indicates the relationship with the "past" code $C_{[0,k)}$ of block length k, which starts at time 0 in the trellis. We will use above notations in the subsequent presentation, which slightly differ from the previous ones, in order to well distinguish the subcodes and state codes that are necessarily defined at more than one subintervals and times.

More specifically, the subcode $C_{[0,k)} \in C$ contains all the codewords that starts from all-zero state at time 0 and ends at all-zero state at time k, while the subcode $C_{[k,N)} \in C$ consists of all the codewords that starts from all-zero state at time k and ends at all-zero state at time N. Let us keep on partitioning the code $C_{[k,N)}$ into subcodes relative to time index $k+j$. Therefore, we have

$$\mathbf{c}_{[k,N)}=\mathbf{c}_{[k,k+j)}+\mathbf{s}_{k,j}+\mathbf{c}_{[k+j,N)} \tag{3-34}$$

where the subscripts of codes $C_{[k,k+j)}$ and $C_{[k+j,N)}$ have the similar meaning as that of the codes $C_{[0,k)}$ and $C_{[k,N)}$, the subscript of the codeword of state code $\mathbf{s}_{k,j} \in S_{k,j}$ indicates the relationship with the "past" code $C_{[k,k+j)}$ of block length j, which starts at time k in the trellis.

Substituting (3-34) into (3-33), it yields

$$\begin{aligned}\mathbf{c} &= \mathbf{c}_{[0,k)} + \mathbf{s}_{0,k} + (\mathbf{c}_{[k,k+j)} + \mathbf{s}_{k,j} + \mathbf{c}_{[k+j,N)}) \\ &= \mathbf{c}_{[0,k)} + (\mathbf{s}_{0,k} + \mathbf{c}_{[k,k+j)} + \mathbf{s}_{k,j}) + \mathbf{c}_{[k+j,N)}\end{aligned} \tag{3-35}$$

It is clearly from (3-33) and (3-34) that the state code $S_{0,k}$ is relevant with the subcodes $C_{[0,k)}$ and $C_{[k,N)}$ with respect to time k, and the state code $S_{k,j}$ is in relation to the subcodes $C_{[k,k+j)}$ and $C_{[k+j,N)}$ with respect to time $k+j$. Actually, from (3-35) the generalized linear space $B_{k,j}$ in (3-32) becomes

$$B_{k,j} = \{(\mathbf{s}_{0,k} + \mathbf{c}_{[k,k+j)} + \mathbf{s}_{k,j})\} \tag{3-36}$$

Since codes $C_{[0,k)}$ and $C_{[k+j,N)}$ are chosen independently, thus we have

$$\dim B_{k,j} = \dim C - \dim C_{[0,k)} - \dim C_{[k+j,N)} \tag{3-37}$$

Clearly, we can obtain the dimension of the branch space by letting j=1 in (3-37), i.e.,

$$\dim B_{k,1} = \dim C - \dim C_{[0,k)} - \dim C_{[k+1,N)} \tag{3-38}$$

Moreover, (3-37) is equivalent to (3-3) as the evaluation of the dimension of state code S_k at the time k if we simply set j=0. Therefore, we can finally form an overall process of

generalization from the state code S_k to the generalized linear space $B_{k,j}$, which includes the branch space $B_{k,1}$ as a special case.

Example 3.7: Let us consider the (8, 4, 4) RM code with P={0, 1} again, whose generator matrix is given by (3-20) and two-section trellis is shown in Fig. 3.7. Now we will keep on partitioning the code $C_{[2,8)}$ into two subcodes $C_{[2,6)}$ and $C_{[6,8)}$, which are a (4, 1, 4) code with generator matrix [1111] and a trivial (2, 0, ∞) code, respectively. The generator matrix of $C_{[2,8)}$ is

$$G=\begin{bmatrix}1 & 1 & 1 & 1 & 0 & 0\\ 0 & 0 & 1 & 1 & 1 & 1\end{bmatrix} \tag{3-39}$$

which is virtually a (6, 2, 4) code defined by the submatrix in (3-20). The state code, which is relative to the partition of $C_{[2,8)}$ by (4, 1, 4) and (2, 0, ∞) codes, is a (6, 1, 4) code with the generator matrix [001111] as given by the second row vector in (3-39).

Hence, the subcode $C_{[2,8)}$ in the trellis representation in Fig. 3.7, which is from the zero state at time k=2 to the zero state at the end, can be further decomposed as follows:

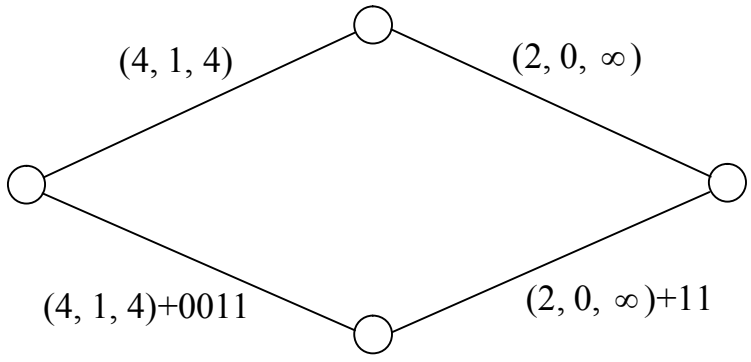


Fig. 3.10 The two-section trellis representation for a (6, 2, 4) subcode $C_{[2,6)}$ in the (8, 4, 4) RM code.

Finally, by the generator matrix in (3-20) the (8, 4, 4) RM code can be expressed by coset decomposition in a three-section trellis in Fig. 3.11.

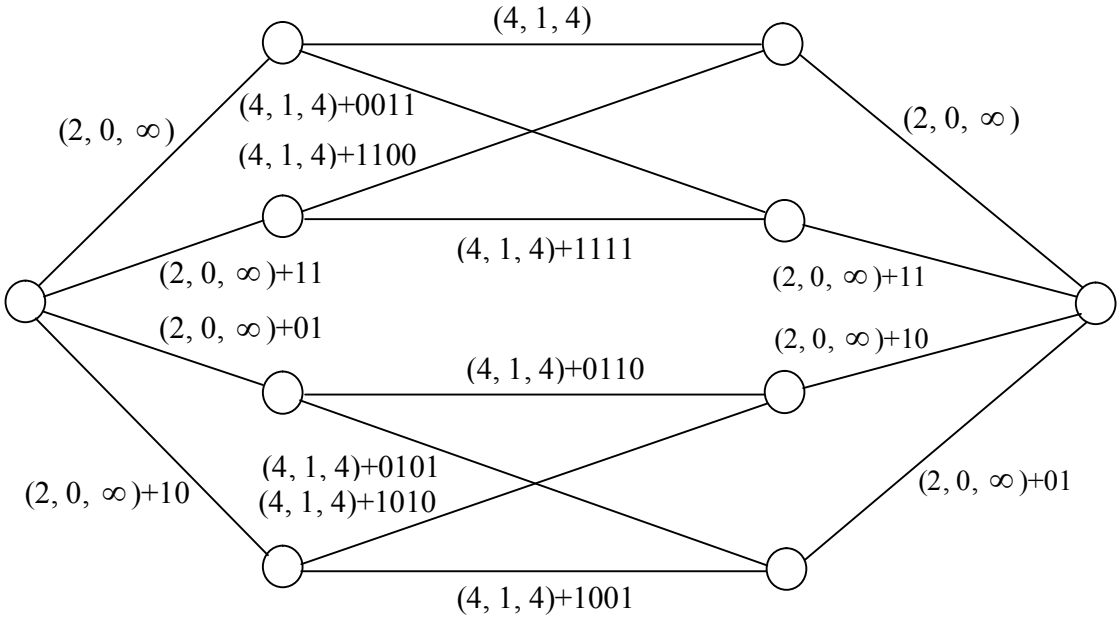


Fig. 3.11 The three-section trellis for the (8, 4, 4) Reed-Muller code by coset decomposition.

From (3-35) and the generator matrix in (3-20), we conclude that any codeword $\mathbf{c} \in$(8, 4, 4) RM code can be uniquely expressed as

$$\begin{aligned}\mathbf{c} &= \mathbf{c}_{[0,2)} + (\mathbf{s}_{0,2} + \mathbf{c}_{[2,6)} + \mathbf{s}_{2,4}) + \mathbf{c}_{[6,8)} \\ &= \mathbf{s}_{0,2} + \mathbf{c}_{[2,6)} + \mathbf{s}_{2,4}\end{aligned} \tag{3-40}$$

where $\mathbf{s}_{0,2}$, $\mathbf{c}_{[2,6)}$ and $\mathbf{s}_{2,4}$ are codewords of the state code $S_{0,2}$, subcode $C_{[2,6)}$ and state code $S_{2,4}$, respectively, with generator matrices given as follows:

$$G_{0,2}=\begin{bmatrix}1 & 1 & 1 & 1 & 0 & 0 & 0 & 0\\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0\end{bmatrix} \tag{3-41a}$$

$$G_{[2,6)}=\begin{bmatrix}0 & 0 & 1 & 1 & 1 & 1 & 0 & 0\end{bmatrix} \tag{3-41b}$$

$$G_{2,4}=\begin{bmatrix}0 & 0 & 0 & 0 & 1 & 1 & 1 & 1\end{bmatrix} \tag{3-41c}$$

Note that the block length of the subcode and state codes are all set to be eight in order to agree with the component-wise modulo-2 addition in (3-40). Clearly, the dimension of the generalized linear space $B_{2,4}$ is 4 because the subcodes $C_{[0,2)}$ and $C_{[6,8)}$ are all trivial with zero dimension.

3.3 General Linear Behavioral Realizations

It is well know that the linear block codes by generator, parity-check matrix and trellis realizations are all special cases of a general class of realizations called behavioral realizations, which is pioneered by Willems from behavioral approach to system theory [15].

In this section we will only consider linear behavioral realizations, where the variables are over a field and the constraints are linear. In the simplest case, the variables are field elements and the constraints are linear equations involving the variables. In the general case, the variables can be vector spaces over the filed, and the constraints are expressed in terms of linear codes, called constraint codes that will be introduced subsequently.

3.3.1 Elementary Linear Behavioral Realization of Block Codes

The elements of a general elementary linear realization of a linear (n, k, d) block code over finite filed $GF(q)$, where $q = p^m$ and p is a prime number, are as follows,

(1) The n codeword symbols $\mathbf{y}=\{ y_i \in GF(q),\ i \in \mathrm{I}\}$, where I denotes the symbol index set. We will use variable **y** instead of **c** for codeword symbol in the section in order to clearly distinguish it from the notation of codeword.

(2) An additional set of s auxiliary variables $\mathbf{s}=\{ s_j \in GF(q),\ j \in \mathrm{J}\}$, where J denotes the state variable index set. The state variables are often hidden, latent and unobserved.

(3) A set of e linear equations over $GF(q)$ involving the components of the symbol and state variables, called the constraint equations.

Note that an ordinary binary code is resulted in if p=2 and m=1. The full behavior B generated by the realization is the set of all combinations or trajectories (**y**, **s**) of symbol and state variables that satisfy all constraint equations. The code C generated by the realization is the set of all symbol n-tuples **y** that appear in any trajectory $(\mathbf{y}, \mathbf{s}) \in B$, i.e., such that there exists some set **s** of state variables such that $(\mathbf{y}, \mathbf{s}) \in B$.

In general, the e linear constraints equations may be written in matrix form as

$$\mathbf{y}A_1 + \mathbf{s}A_2 = 0 \tag{3-42}$$

where **y** is a row n-tuples of symbol vectors, **s** is a row s-tuples of state variable components, and A_1 and A_2 are $n \times e$ and $s \times e$ matrices over $GF(q)$, respectively. Clearly,

the set B of all solutions (**y**, **s**) to such a set of equations is a subspace of the vector space $\left(GF(q)\right)^{n+s}$, and the code C is the projection of B onto its first n components.

We will show by the following examples that generator matrix and parity-check matrix yield elementary behavioral realizations of this kind.

Example 3.8 (generator realizations): Let G be a $k\times n$ generator matrix for linear block code C, whose k rows forms a set of linearly independent generators for C. Then C is the set of all n-tuples of the form $\mathbf{y}=\mathbf{u}G$ for some information k-tuples $\mathbf{u}\in\left(GF(q)\right)^k$. Thus C has an elementary linear behavioral realization with a symbol n-tuples **y** and a state k-tuple **u** and n constraint equations, namely,

$$\mathbf{y}-\mathbf{u}G=\mathbf{0} \tag{3-43}$$

Using the trellis-oriented generator (3-31) for the (8, 4, 4) RM code, we obtain a linear behavioral realization with 8 symbol variable, 4 state variable and 8 constraint equations over GF(2) as follows

$$y_0=u_0 \tag{3-44a}$$

$$y_1=u_0+u_1 \tag{3-44b}$$

$$y_2=u_0+u_2 \tag{3-44c}$$

$$y_3=u_0+u_1+u_2 \tag{3-44d}$$

$$y_4=u_1+u_2+u_3 \tag{3-44e}$$

$$y_5=u_2+u_3 \tag{3-44f}$$

$$y_6 = u_1 + u_3 \tag{3-44g}$$

$$y_7 = u_3 \tag{3-44h}$$

Example 3.9 (parity-check realizations): Let H be an $(n-k)\times n$ parity-check matrix for linear block code C. Then C is the set of all n-tuples that satisfies the $n-k$ constraint equations

$$H\mathbf{y}^T = \mathbf{0}^T \tag{3-45}$$

This corresponds to an elementary linear behavioral realization with no state variable.

Since the (8, 4, 4) RM code C is self-dual, the trellis-oriented generator matrix is also a parity-check matrix for C. It yields an elementary linear behavioral realization with no state variables, 8 symbol variables and 4 constraint equations over GF(2) as follows

$$y_0 + y_1 + y_2 + y_3 = 0 \tag{3-46a}$$

$$y_1 + y_3 + y_4 + y_6 = 0 \tag{3-46b}$$

$$y_2 + y_3 + y_4 + y_5 = 0 \tag{3-46c}$$

$$y_4 + y_5 + y_6 + y_7 = 0 \tag{3-46d}$$

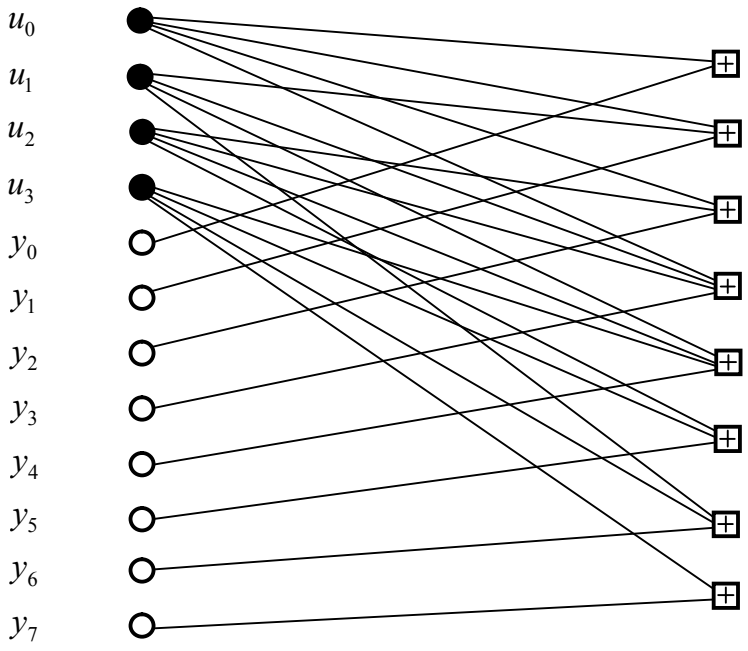
3.3.2 Graphs of Elementary Linear Behavioral Realizations

In coding theory, a graph that describes an elementary linear behavioral realization is called a Tanner graph that is introduced in Chapter 2.

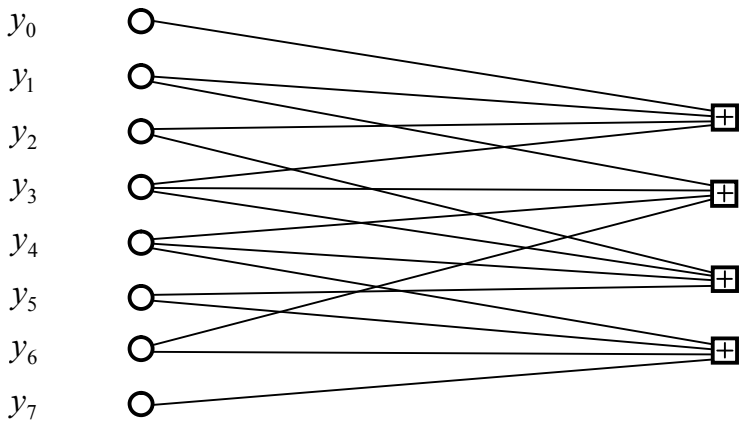
(1) **Bipartite graph for linear behavioral realizations**: Such a graph has two types of vertices, namely $n+s$ vertices corresponding to the n symbol and s state variables, and e

vertices corresponding to the e constraint equations. An edge is drawn be between a variable vertex and a constraint vertex if the corresponding variable is involved in the corresponding constraint. Thus, the graph is bipartite, i.e., the vertices are partitioned into two sets such that every edge connects a vertex of one type to one of the other type.

Example 3.10: The bipartite graphs for the (8, 4, 4) RM code by generator and parity-check realizations are shown in Fig. 3.12, where the symbol and state variables are represented by vertices with open and filled circles, respectively, while the constraint vertices are represented by squares with "+".



(a)



(b)

Fig. 3.12 The bipartite graphs for the (8, 4, 4) RM code by (a) generator and (b) parity-check realizations, respectively, with symbol, state and constraint vertices.

Obviously, we may remove the state vertices from the bipartite graph in Fig. 3.12(a) since the information and codeword bits are correlated. This results in a more compact representation by the parity-check realization as shown in Fig. 3.12(b).

Generically, the bipartite graph can be employed to represent any linear code by using the inherit check constraints, which are recognized by the check-sum for each row of the parity-check matrix.

The bipartite graph of parity-check realization is also a basic and effective method to describe the phenomenal low-density parity-check (LDPC) codes that will be introduced in the Chapter 3.

(2) **Branch space**: Let us revisit the branch space B_k, k=0, 1, …, n–1, for a linear block code of length n, whose generalized form is given by (3-32). Actually, B_k presents the local constraint consisting of two consecutive states S_k and S_{k+1} in time axis and a codeword symbol y_k between them. Thus, the branch space B_k is also called as the constraint code at time k, which differs from the check-sum constraints for the rows of parity-check matrix. Therefore, by the parity-check matrix in (3-1) and the eight-section trellis shown in Fig. 3.3, we can present another version of bipartite graph of the (8, 4, 4) RM code illustrated in Fig. 3.13, which includes the constraint code (or branch space) at each time k as well as the symbol/state variables.

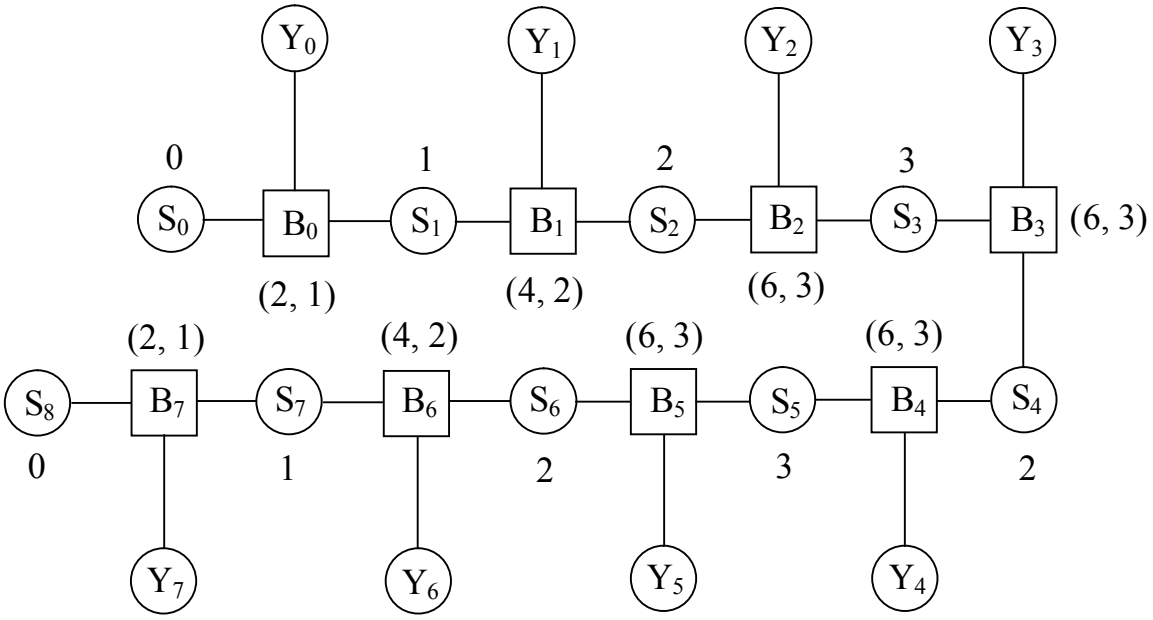


Fig. 3.13. The bipartite graph for trellis realization of the (8, 4, 4) code with the dimensions of state space S_k and branch space (constraint code) B_k at each time instance.

Evidently, the dimensions of S_0 and S_8 are zero because the all the trajectories initially start from all-zero state and eventually end at the same state.

Example 3.11: Let us consider the branch spaces $B_1=\{(s_1,\ y_1,\ s_2)\}\subseteq S_1\times Y_1\times S_2$ at time index k=1, $B_2=\{(s_2,\ y_2,\ s_3)\}\subseteq S_2\times Y_2\times S_3$ at k=2 and $B_3=\{(s_3,\ y_3,\ s_4)\}\subseteq S_3\times Y_3\times S_4$ at k=3, respectively.

(1) $B_1=\{(s_1,\ y_1,\ s_2)\}\subseteq S_1\times Y_1\times S_2$ at time index k=1: It can be easily obtained from Fig. 3.3 that

$$B_1=\begin{Bmatrix}0000\underline{0}0000,0000\underline{1}1001\\0001\underline{0}0001,0001\underline{1}1000\end{Bmatrix} \tag{3-47a}$$

where the first and last four bits for each codeword are taken from the state spaces S_1 and S_2 at k=1 and 2, respectively, and the fifth bit outlined for each codeword is taken from the binary bit set Y_1 between the two time indices. Evidently, the four length-8 vectors in (3-47a) actually form a two-dimensional linear block code. However, by taking its effective length into account, it is treated as effectively a (4, 2, 2) binary constraint code, namely,

$$B_1=\begin{Bmatrix}0000,\ \ 0111\\1001,\ \ 1110\end{Bmatrix} \tag{3-47b}$$

where the generators in the generator matrix can be chosen as

$$G_1=\begin{bmatrix}0111\\1001\end{bmatrix} \tag{3-47c}$$

(**2**) $B_2=\{(s_2, y_2, s_3)\}\subseteq S_2\times Y_2\times S_3$ **at time index *k*=2**: From Fig. 3.3 we have

$$B_2=\begin{Bmatrix}0000\underline{0}0000,0000\underline{1}0011,\\ 0001\underline{0}0001,0001\underline{1}0010,\\ 1000\underline{0}1000,1000\underline{1}1011,\\ 1001\underline{0}1001,1001\underline{1}1010\end{Bmatrix} \tag{3-48a}$$

Similarly as the last case for B_1, the branch space B_2 is regarded as effectively a (6, 3, 2) binary constraint code as

$$B_2=\begin{Bmatrix}000000,001011,\\ 010001,011010,\\ 100100,101111,\\ 110101,111110\end{Bmatrix} \tag{3-48b}$$

where the generator matrix is

$$G_2=\begin{bmatrix}001011\\ 010001\\ 100100\end{bmatrix} \tag{3-48c}$$

(**3**) $B_3=\{(s_3, y_3, s_4)\}\subseteq S_3\times Y_3\times S_4$ **at time index *k*=3**: From Fig. 3.3 we obtain

$$B_3=\begin{Bmatrix}0000\underline{0}0000,0001\underline{1}1010,\\ 0010\underline{0}0010,0011\underline{1}1000,\\ 1000\underline{0}1000,1001\underline{1}0010,\\ 1010\underline{0}1010,1011\underline{1}0000\end{Bmatrix} \tag{3-49a}$$

which can be effectively viewed as a (6, 3, 2) binary constraint code as

$$B_3=\begin{Bmatrix}000\underline{0}00,001\underline{1}11,\\ 010\underline{0}01,011\underline{1}10,\\ 100\underline{0}10,101\underline{1}01,\\ 110\underline{0}11,111\underline{1}00\end{Bmatrix} \tag{3-49b}$$

where the generator matrix is

$$G_3 = \begin{bmatrix} 010001 \\ 011110 \\ 100010 \end{bmatrix} \tag{3-49c}$$

Note that the bipartite graph, represented by the constraint codes as well as the symbol/state variables, is virtually based on the ordinary trellis for a linear block code, but can offer more additional insight details for the trellis representation.