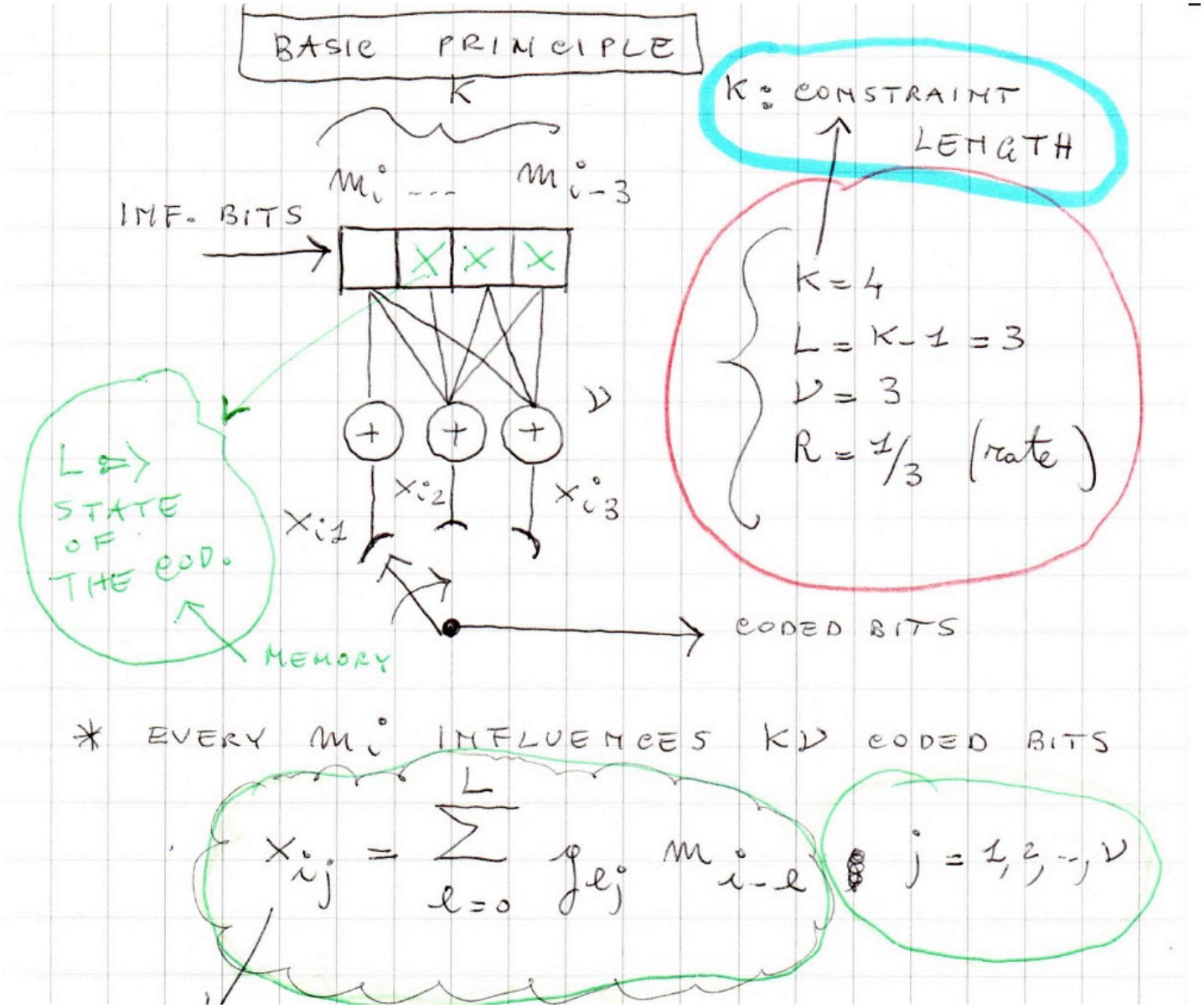
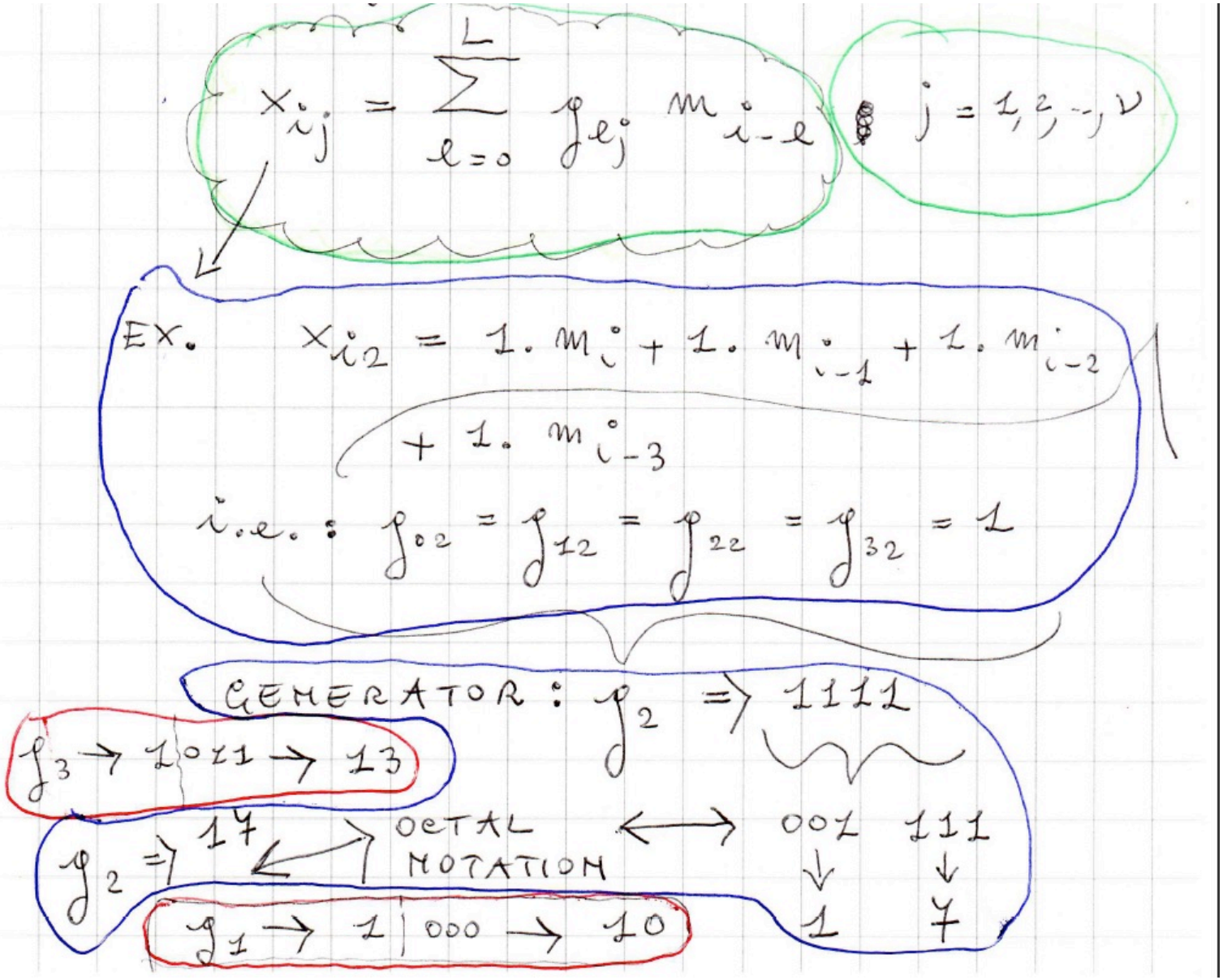
Convolutional Codes

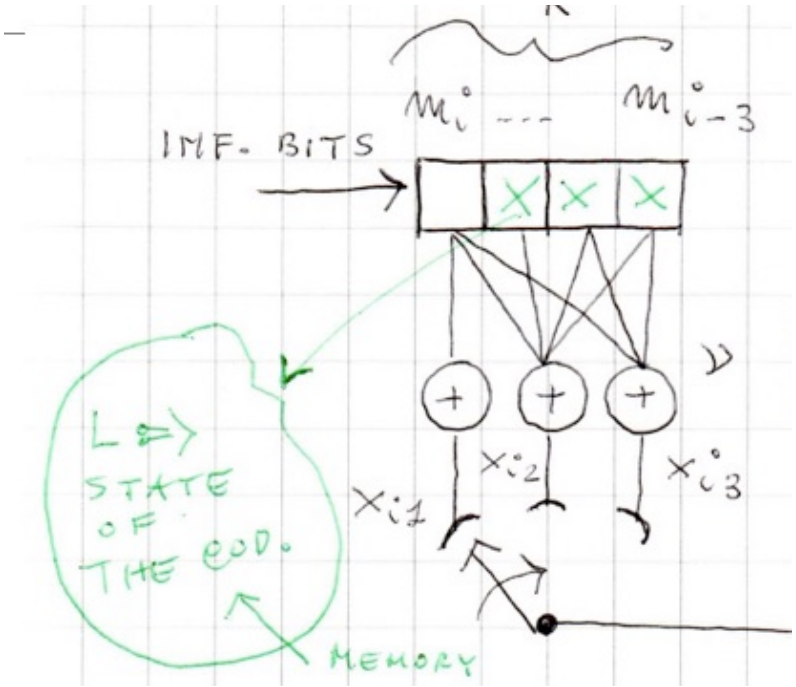
Pierangelo Migliorati

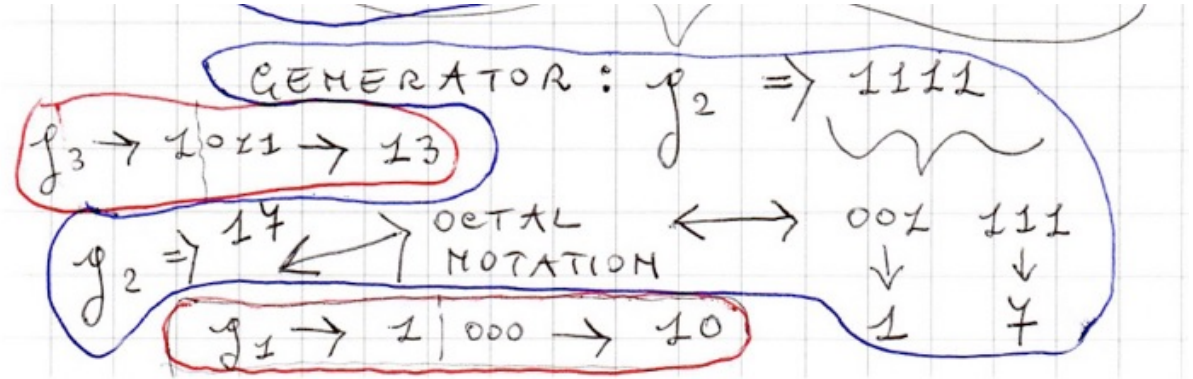
DII - University of Brescia



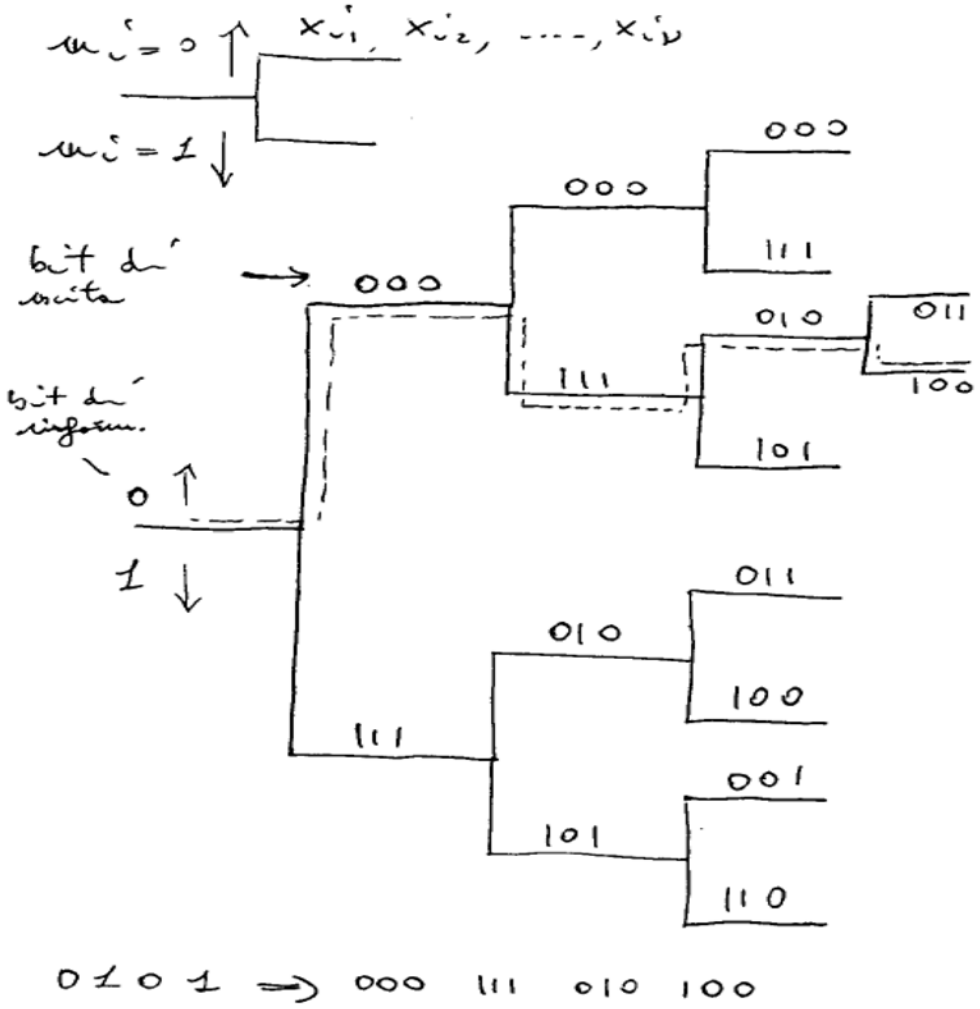


Basic principle: generators

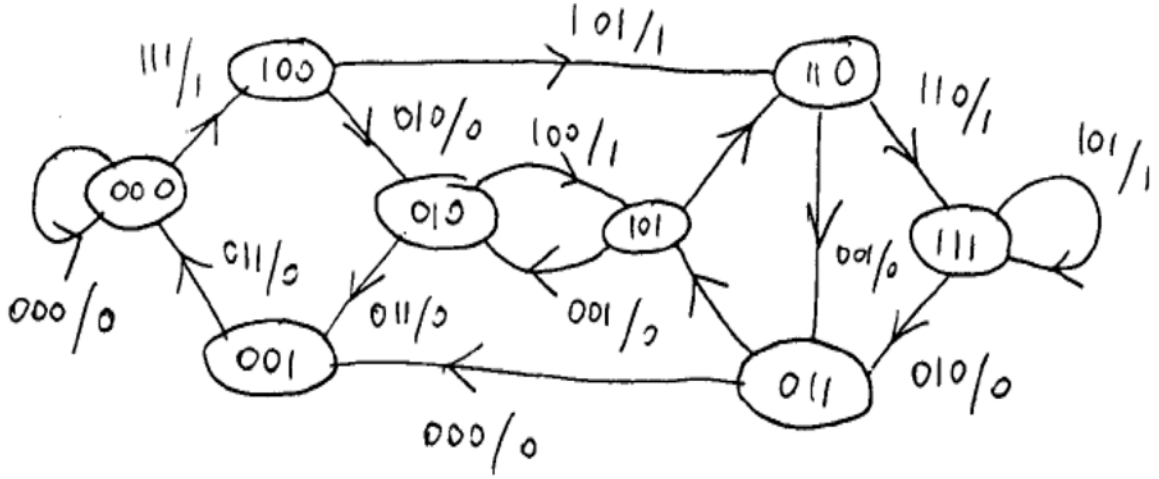


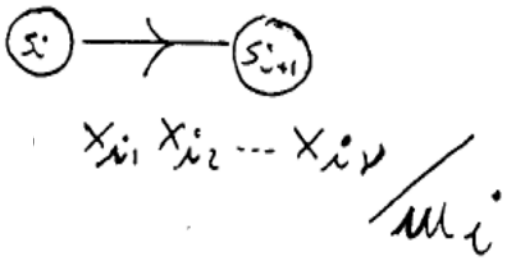


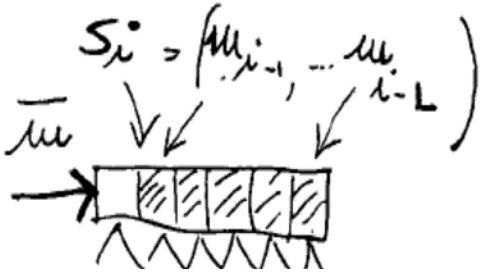
Tree diagrams



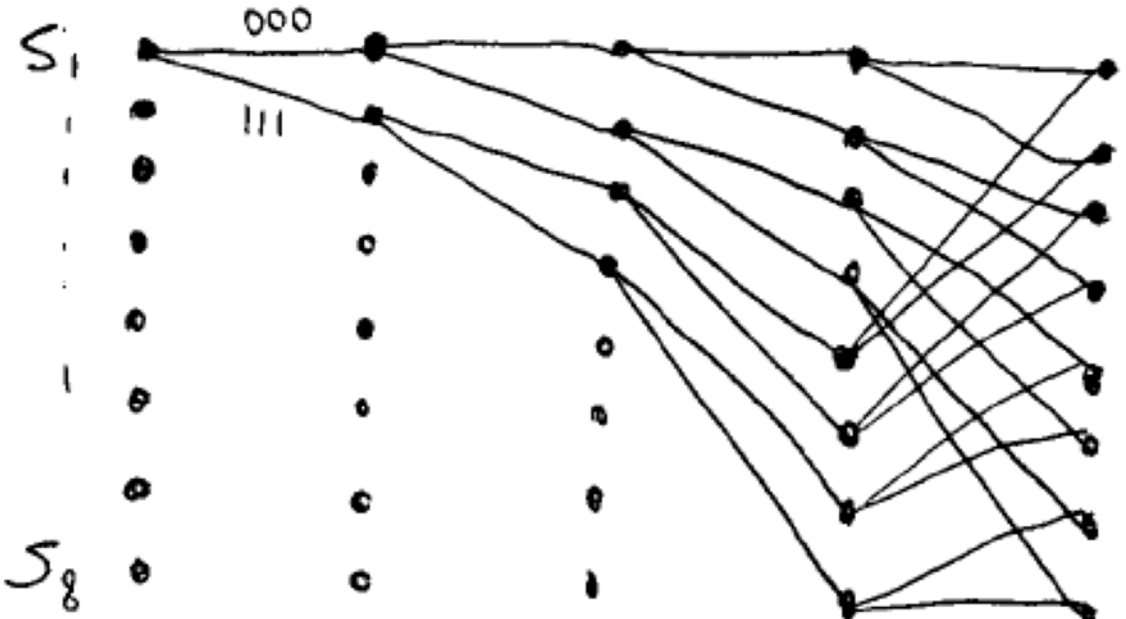
State diagrams

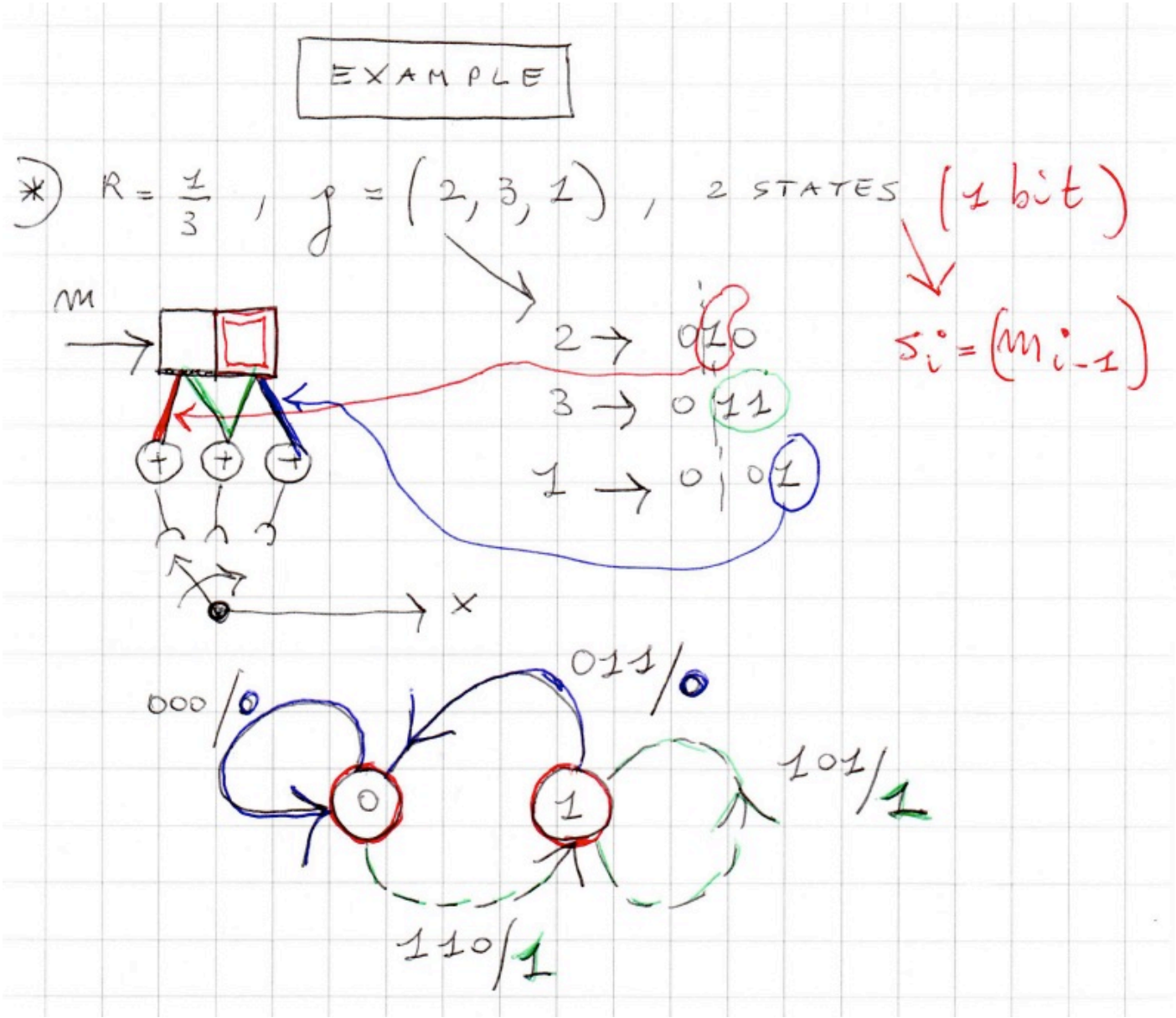




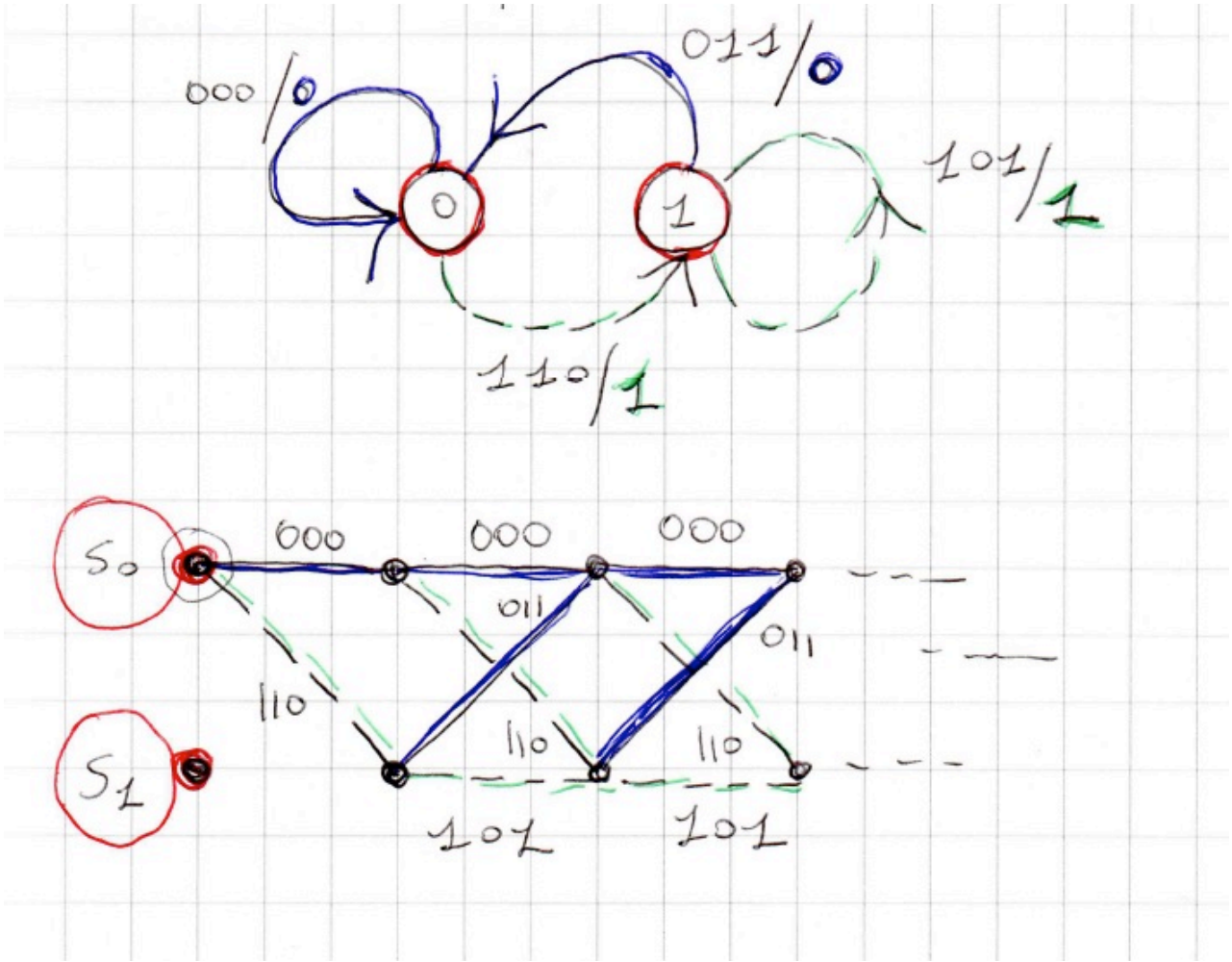


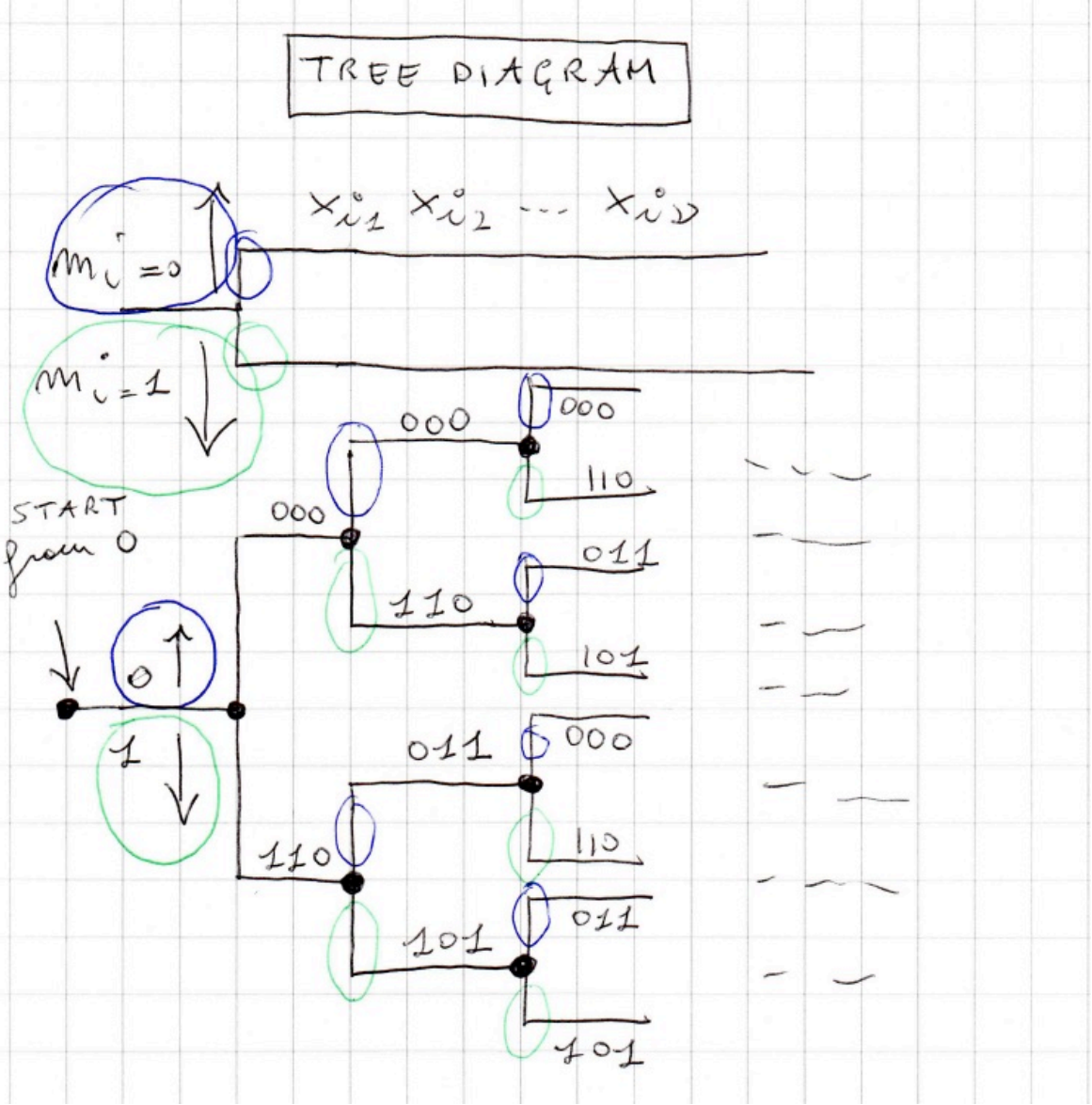
Trellis diagrams

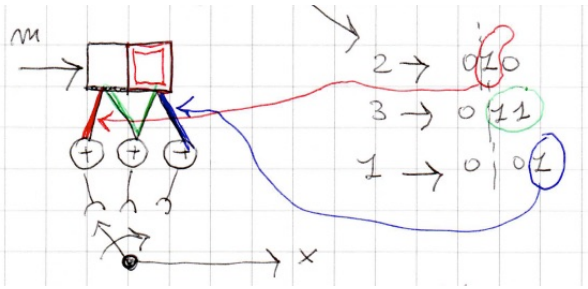


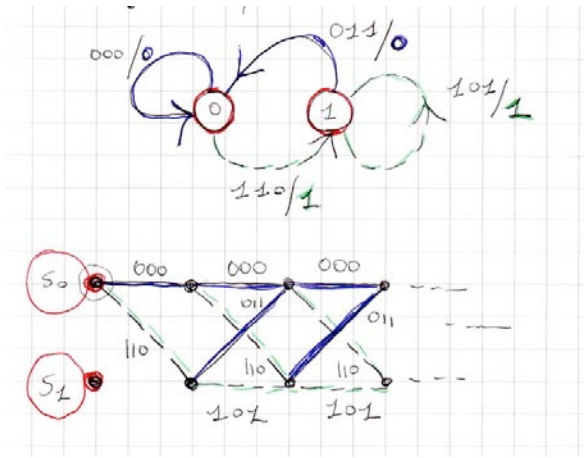


Example



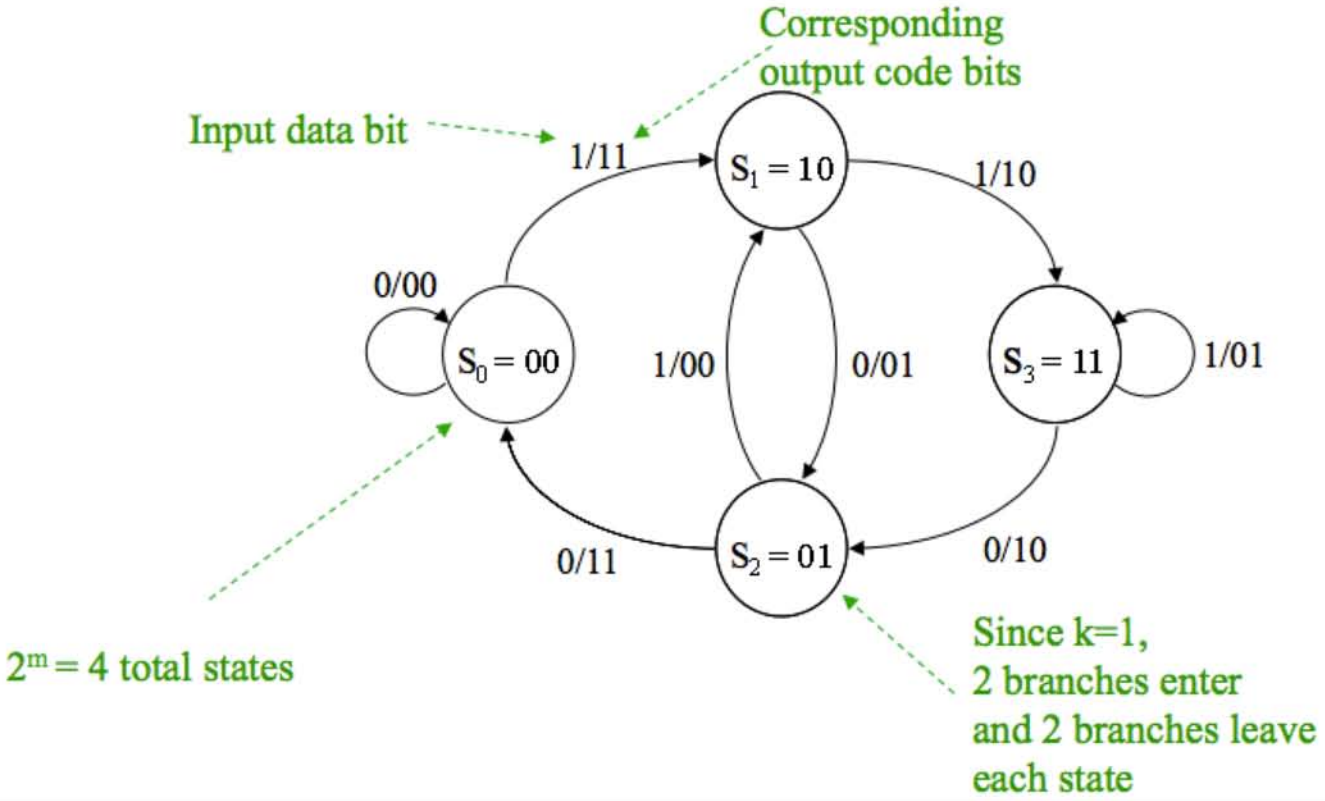






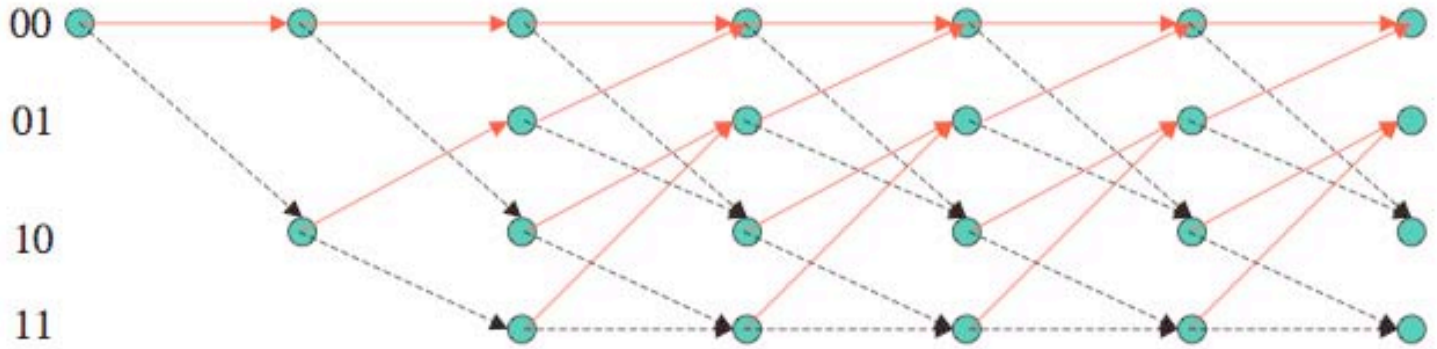
State Diagrams

- A convolutional encoder is a finite state machine, and can be represented in terms of a state diagram.



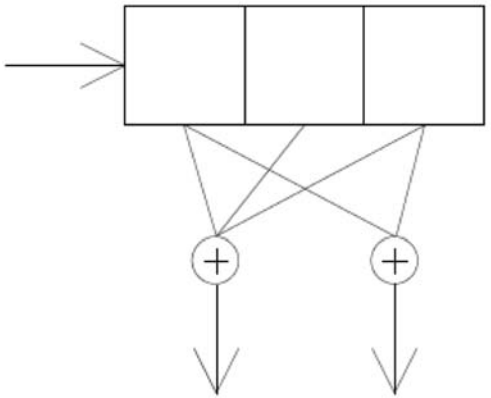
Trellis diagram (2)

Trellis Diagram



Viterbi algorithm used to find the best path through the trellis

Polinomial description (Zeta transform)

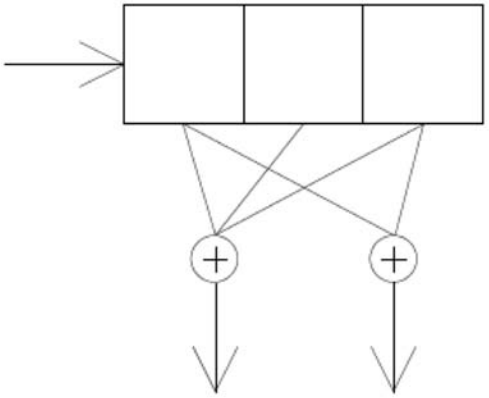


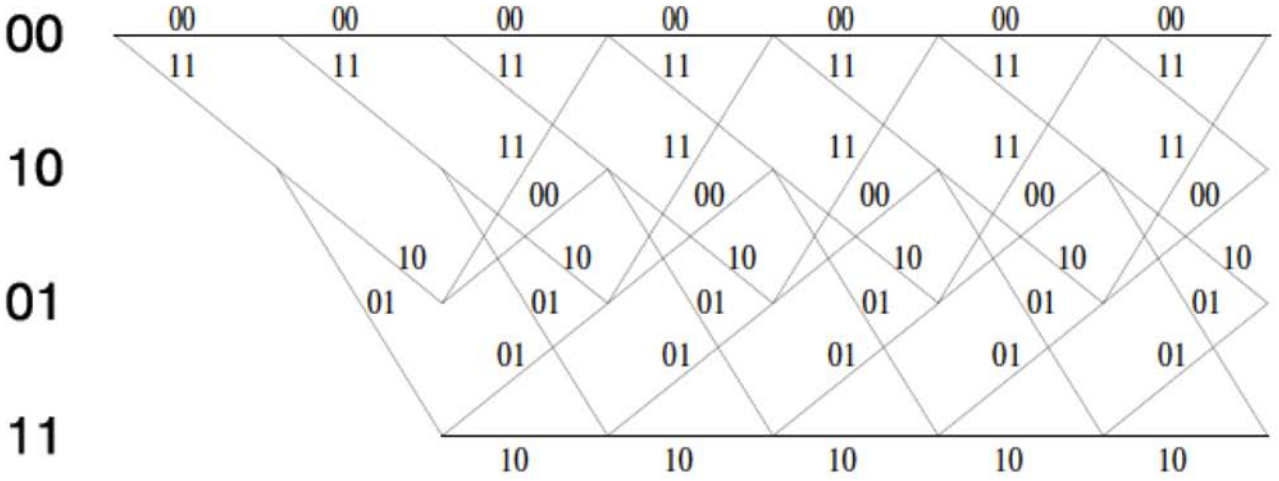
$$i(D) = i_0 + i_1 D + i_2 D^2 + \dots$$

$$c_1(D) = i(D)(1 + D + D^2)$$

$$c_2(D) = i(D)(1 + D^2)$$

Examples





Other Examples

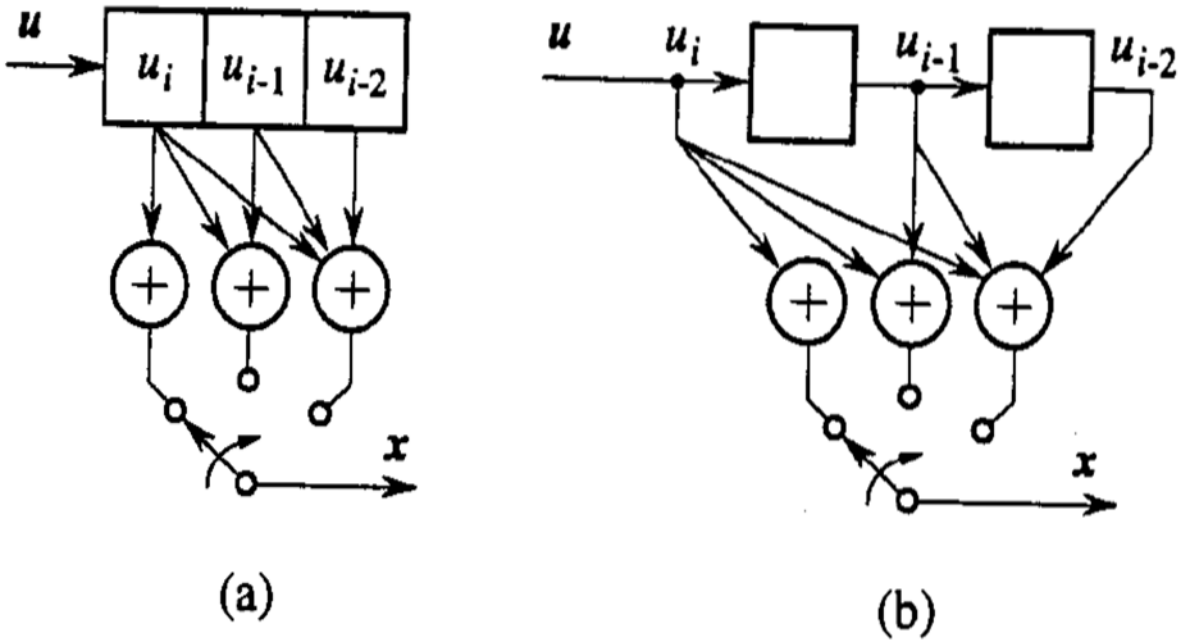


Figure 11.2: *Two equivalent schemes for the convolutional encoder of the (3,1,3) code of Example 11.1.*

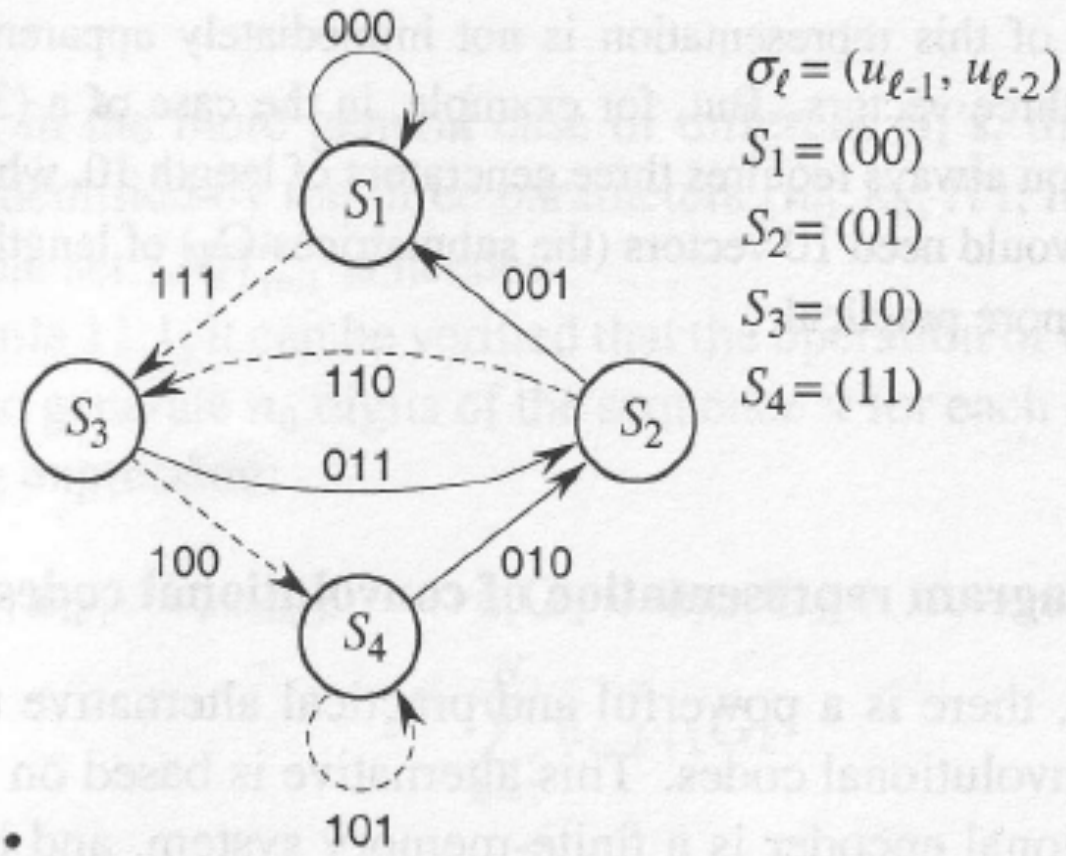


Figure 11.6: *State diagram for the (3,1,3) convolutional code of Example 11.1.*

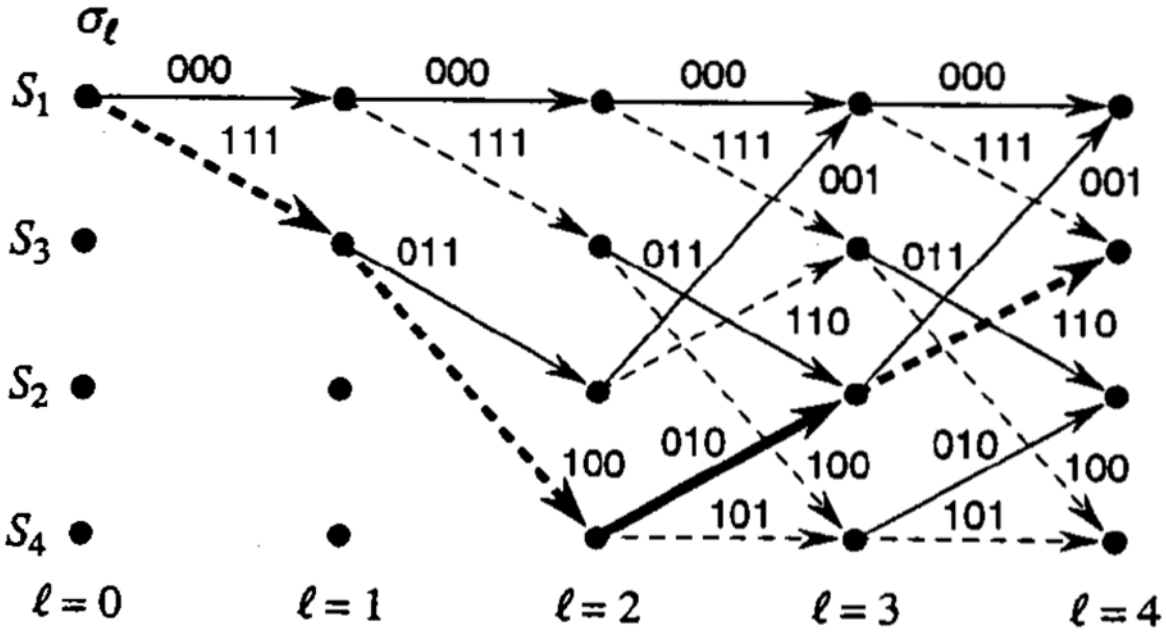


Figure 11.7: *Trellis diagram for the (3,l,3) convolutional code of Example 11.1. The boldface path corresponds to the input sequence 1101.*

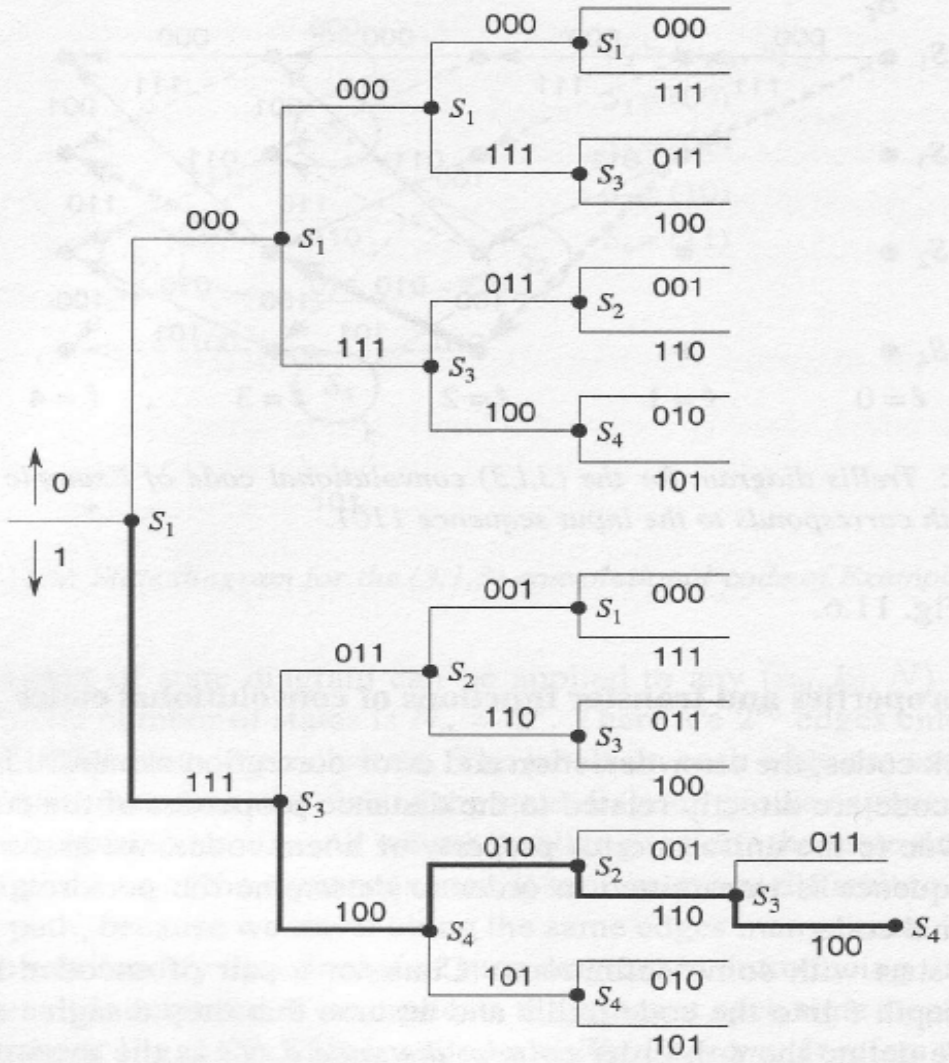
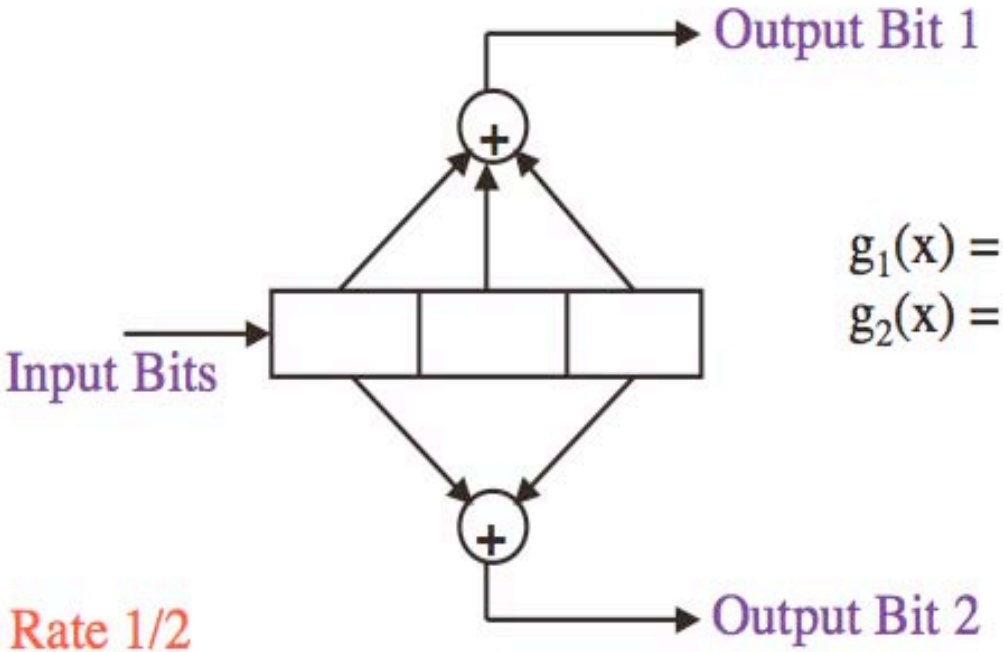


Figure 11.8: *Tree diagram for the (3,1,3) convolutional code of Example 11.1. The solid path corresponds to the input sequence 11011.*

Alternative graphical representations …

Convolutional Codes



$g_1(x) = 1+x+x^2$

$g_2(x) = 1+x^2$

- Rate 1/2
- Encoder has memory
- Non-recursive, non-systematic code

Alternative graphical representations …

$$G^1(D) = 1 + D + D^2$$
$$G^2(D) = 1 + D^2 \qquad [3.13]$$

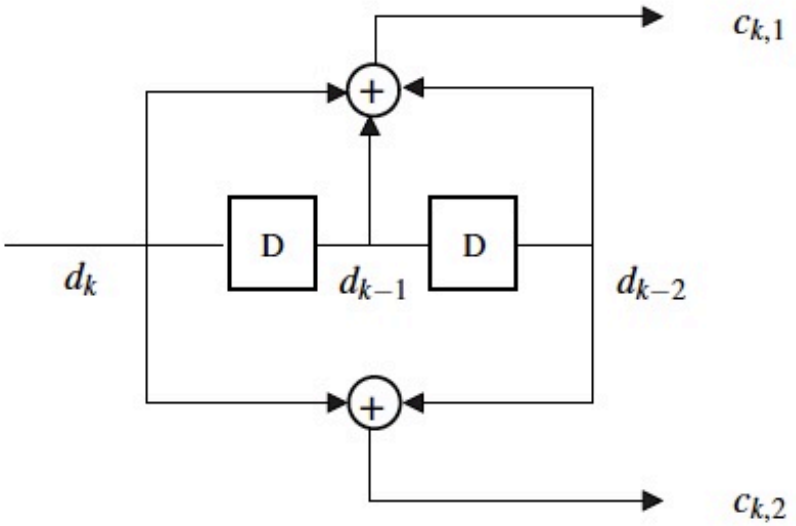


Figure 3.2. *General diagram of a convolution encoder with parameters* $m = 2, R = 1/2$

Alternative graphical representations …

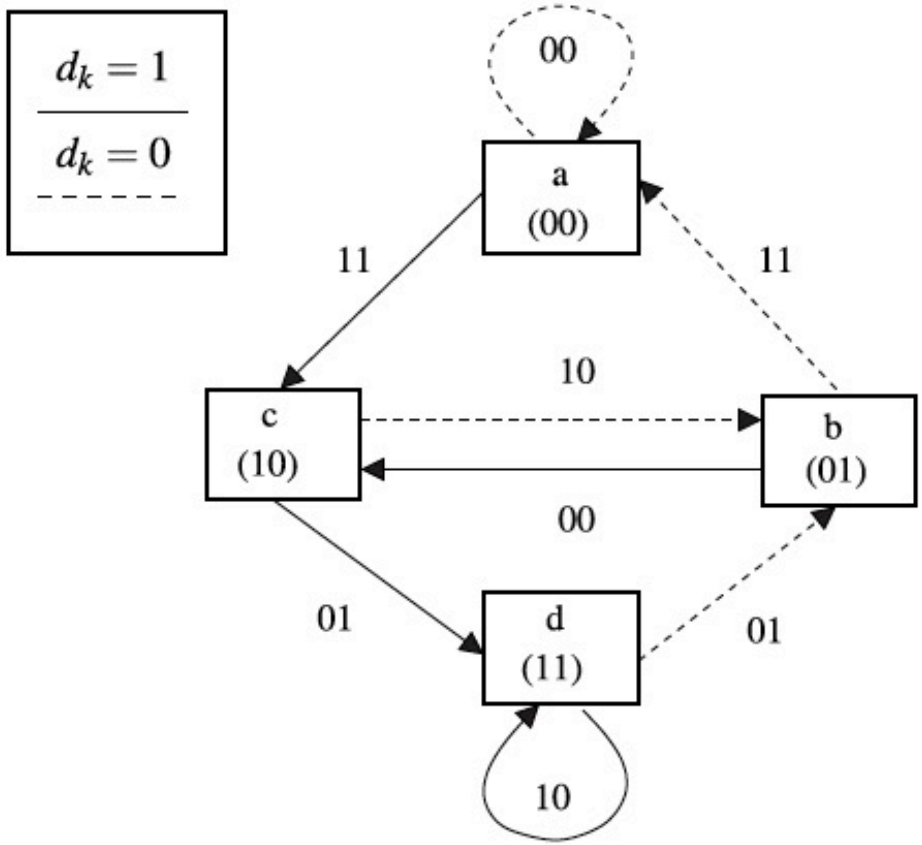


Figure 3.4. *State transition diagram of the convolutional code from example 3.1*

Alternative graphical representations …

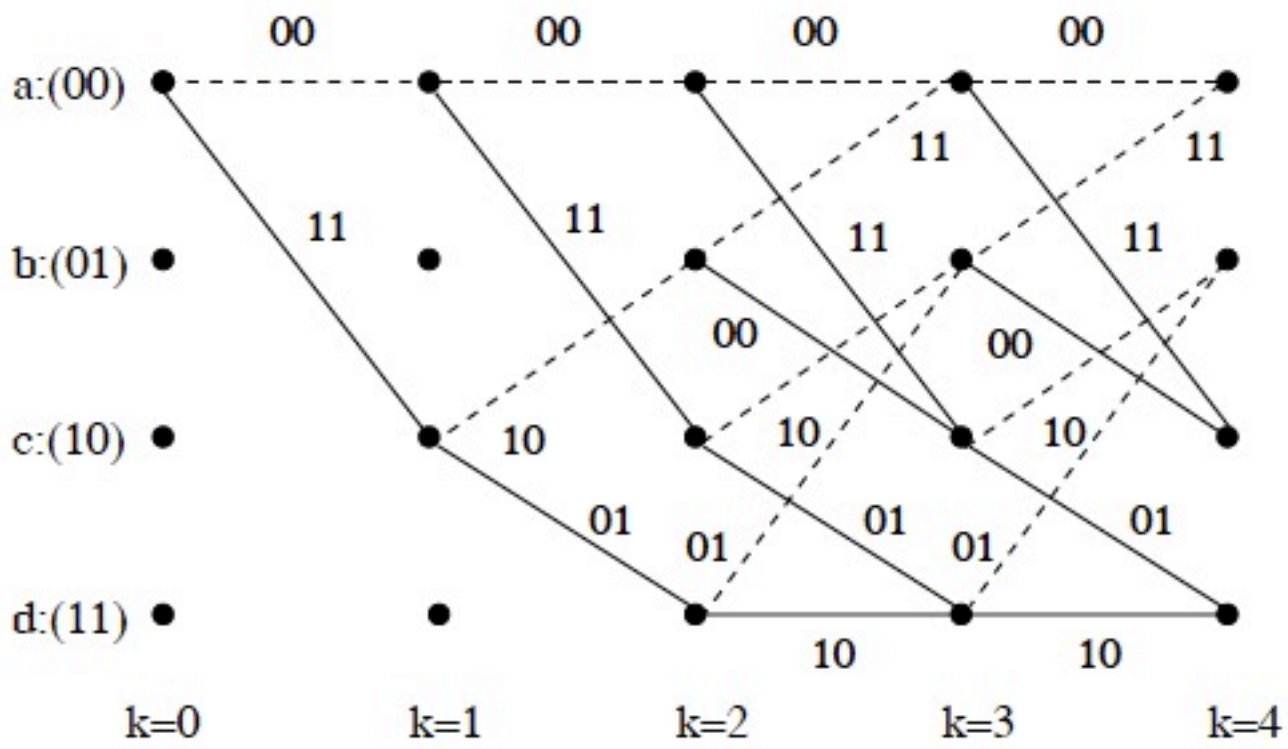


Figure 3.5. *Lattice diagram of the convolutional code from example 3.1*

Alternative graphical representations …

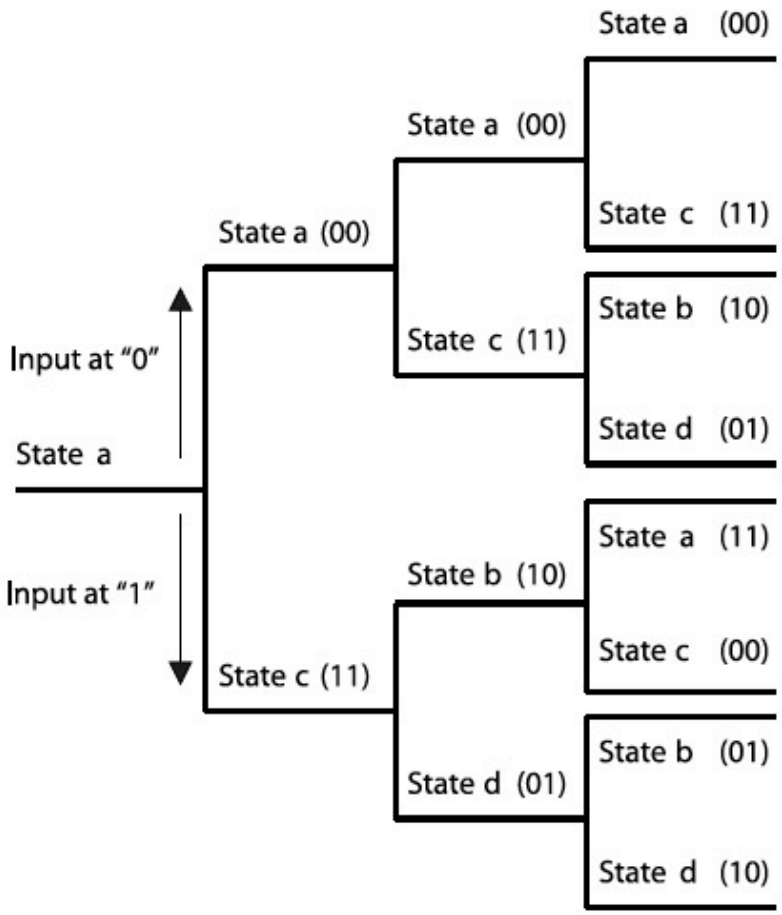


Figure 3.6. *Tree diagram of the convolutional code from example 3.1*

Decoding …

The Viterbi Algorithm

- Walk through the trellis and compute the Hamming distance between that branch of r and those in the trellis.
- At each level, consider the two paths entering the same node and are identical from this node onwards. From these two paths, the one that is closer to r at this stage will still be so at any time in the future. This path is retained, and the other path is discarded.
- Proceeding this way, at each stage one path will be saved for each node. These paths are called the survivors. The decoded sequence (based on MDD) is guaranteed to be one of these survivors.

- Each survivor is associated with a metric of the accumulated Hamming distance (the Hamming distance up to this stage).
- Carry out this process until the received sequence is considered completely. Choose the survivor with the smallest metric.

Decoding

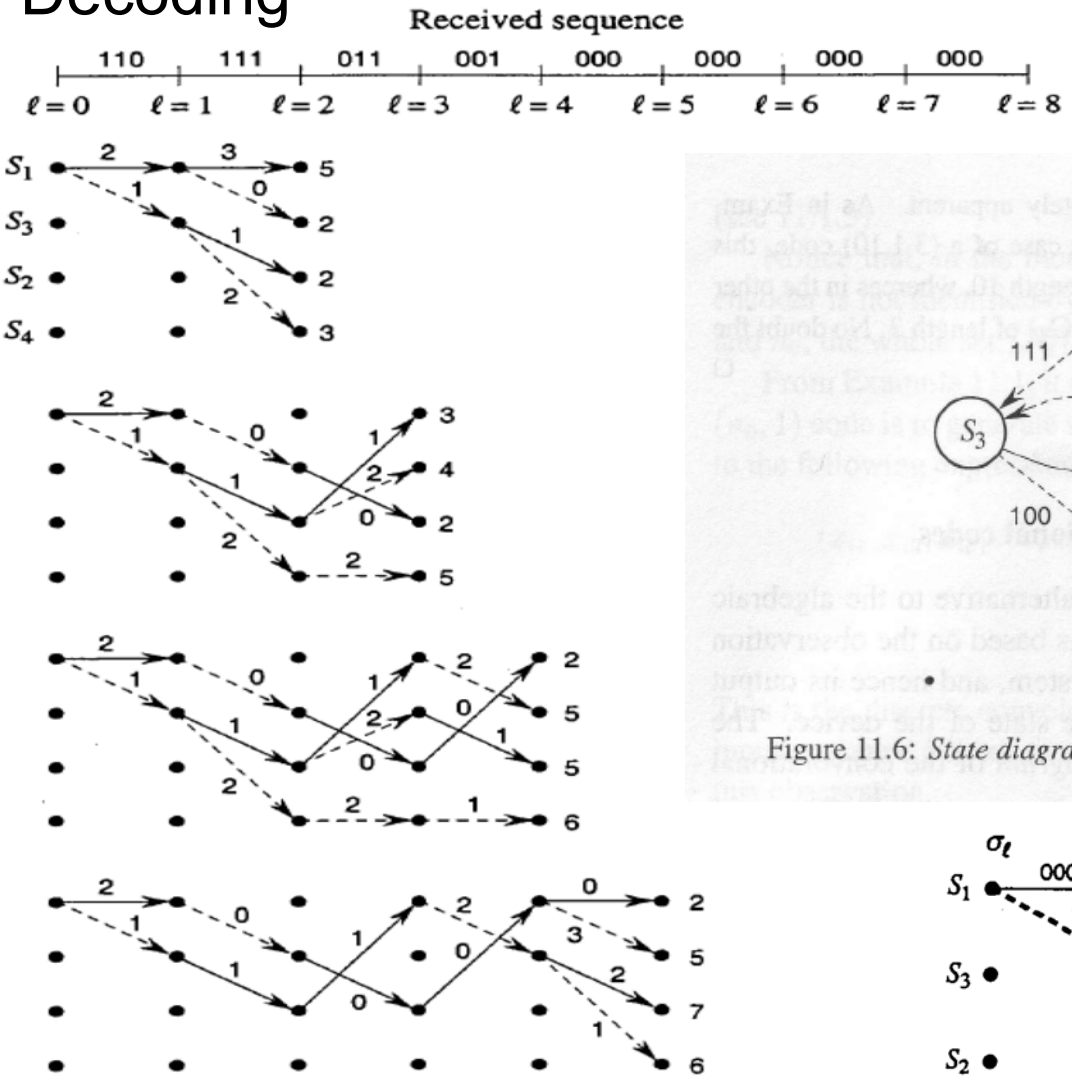


Figure 11.16: *Viterbi decoding algorithm applied to the (3,1,*
Fig. 11.6. The decoded sequence is 01000000.

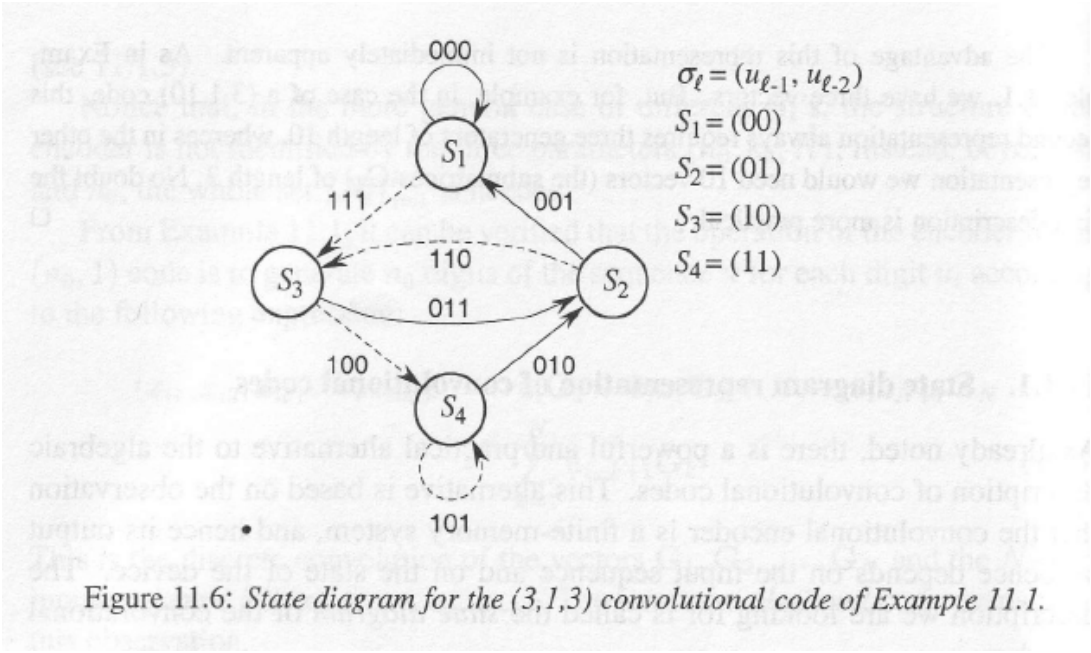


Figure 11.6: *State diagram for the (3,1,3) convolutional code of Example 11.1.*

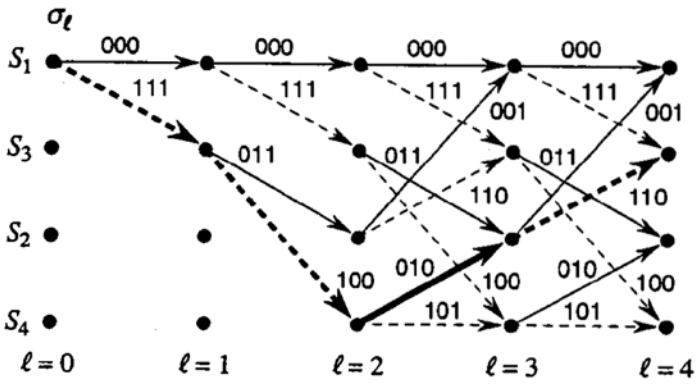


Figure 11.7: *Trellis diagram for the (3,1,3) convolutional code of Example 11.1. The boldface path corresponds to the input sequence 1101.*

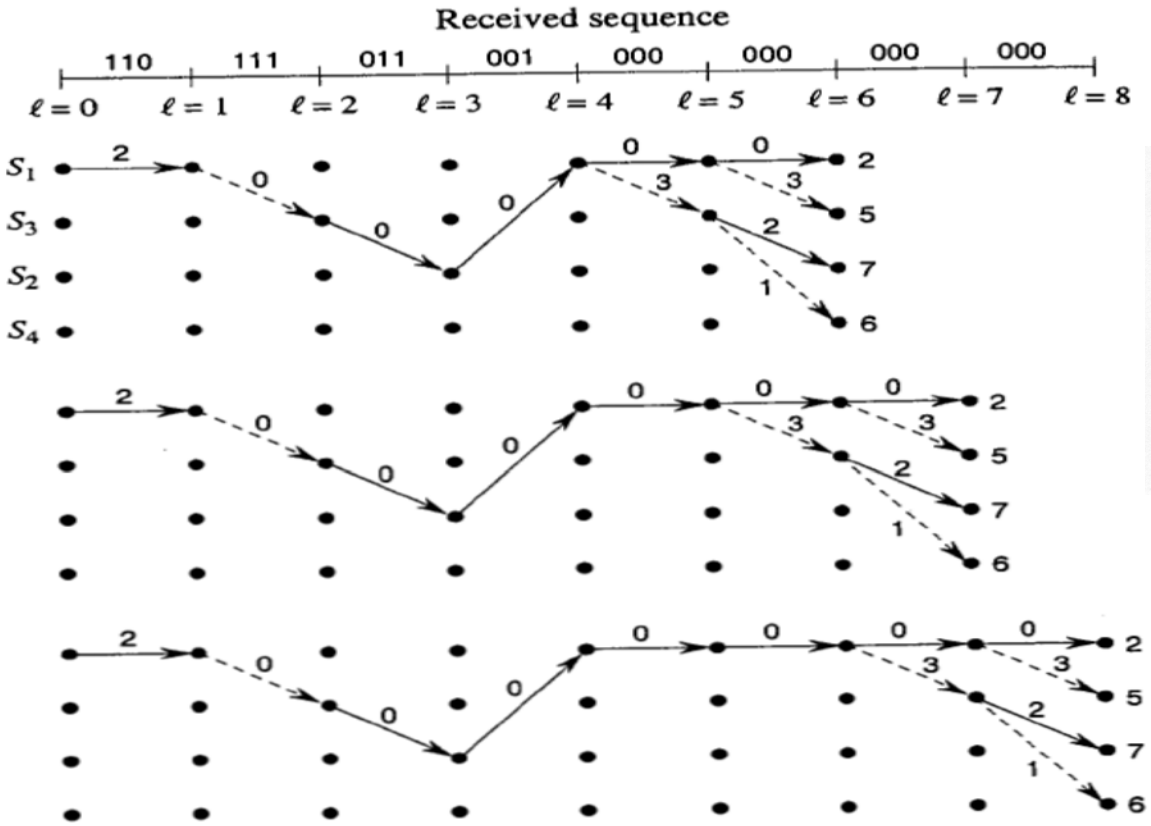


Figure 11.17: *Continuation of Fig. 11.16: Viterbi decoding algorithm applied to the (3,1,3) convolutional code of Fig. 11.6. The decoded sequence is 01000000.*

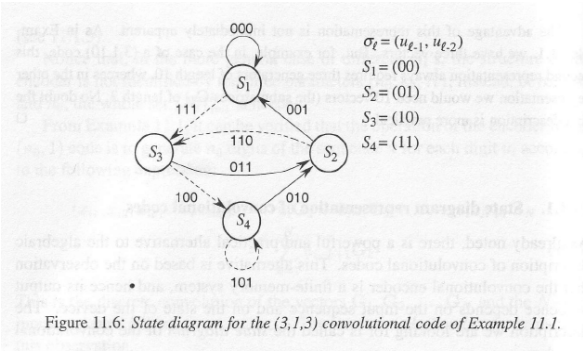


Figure 11.6: *State diagram for the (3,1,3) convolutional code of Example 11.1.*

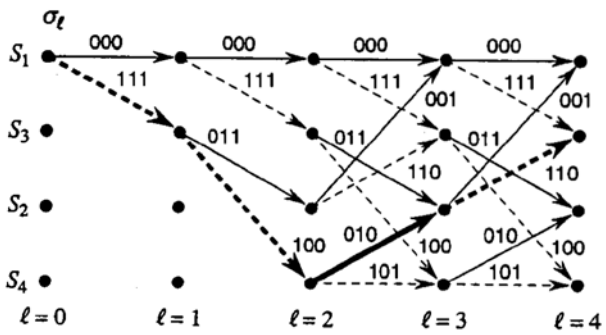


Figure 11.7: *Trellis diagram for the (3,1,3) convolutional code of Example 11.1. The boldface path corresponds to the input sequence 1101.*

Distance Properties of Conv. Codes

- Def: The *free distance*, d_{free}, is the minimum Hamming distance between any two code sequences.
- Criteria for good convolutional codes:
  - Large free distance, d_{free}.
  - Small Hamming distance (i.e. as few differences as possible) between the input information sequences that produce the minimally separated code sequences. d_{inf}
- There is no known constructive way of designing a conv. code of given distance properties. However, a given code can be analyzed to find its distance properties.

- Convolutional codes are linear. Therefore, the Hamming distance between any pair of code sequences corresponds to the Hamming distance between the all-zero code sequence and some nonzero code sequence. Thus for a study of the distance properties it is possible to focus on the Hamming distance between the all-zero code sequence and all nonzero code sequences.
- The nonzero sequence of minimum Hamming weight diverges from the all-zero path at some point and remerges with the all-zero path at some later point.

Probability of error

• A decoding error leads to a wrong path in the trellis

• The error probability over a sequence goes to one as N increases without bounds

• We need to normalize the probability of error to the sequence length. We compute the probability that, at instant *t*, an “error event” begins

• We then consider error events of finite length, that is, paths that deviate from the correct one only for a short period (more probable errors are between words at small length, which are associated to paths with only short deviations)

• The code is linear, so we assume we send the all-zeros word. The correct path in the trellis is the paths that always stays in the first state

• We study error events that start from the first state and end in the first state, from short ones to longer ones

• We compute the number of errors on the information bits for each error event

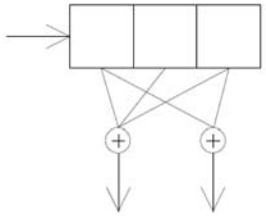
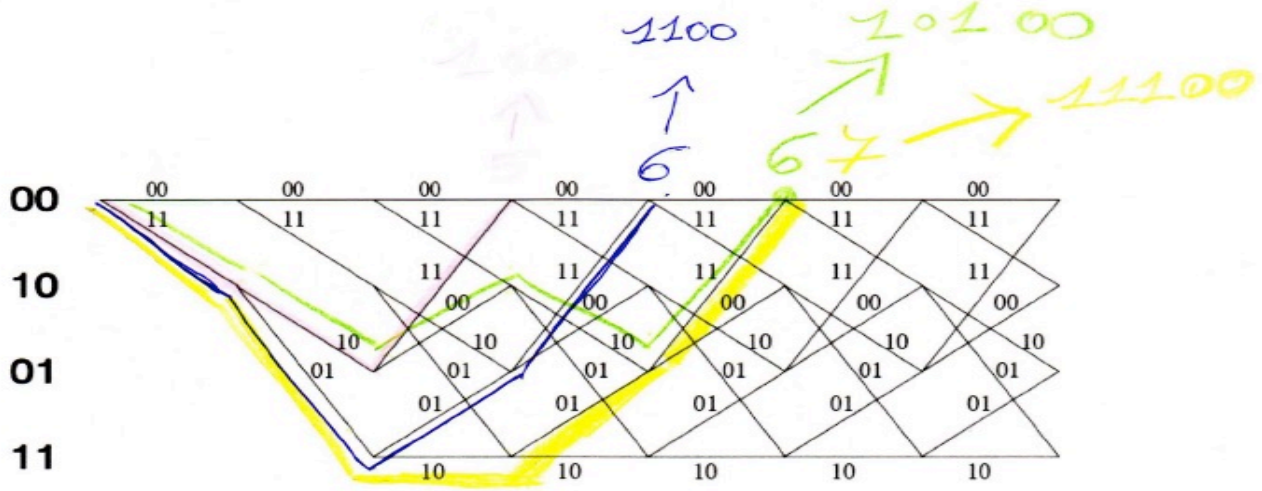


Figura 3.1: Semplice codificatore convoluzionale



L = 3, d = 5, 1 bit

L = 4, d = 6, 2 bits err

L = 5, d = 6, 2 bits ε

L = 5, d = 7, 3 b ε

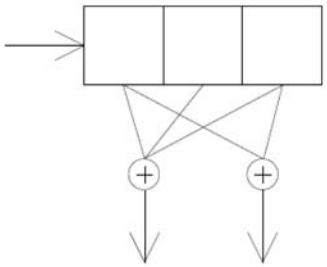
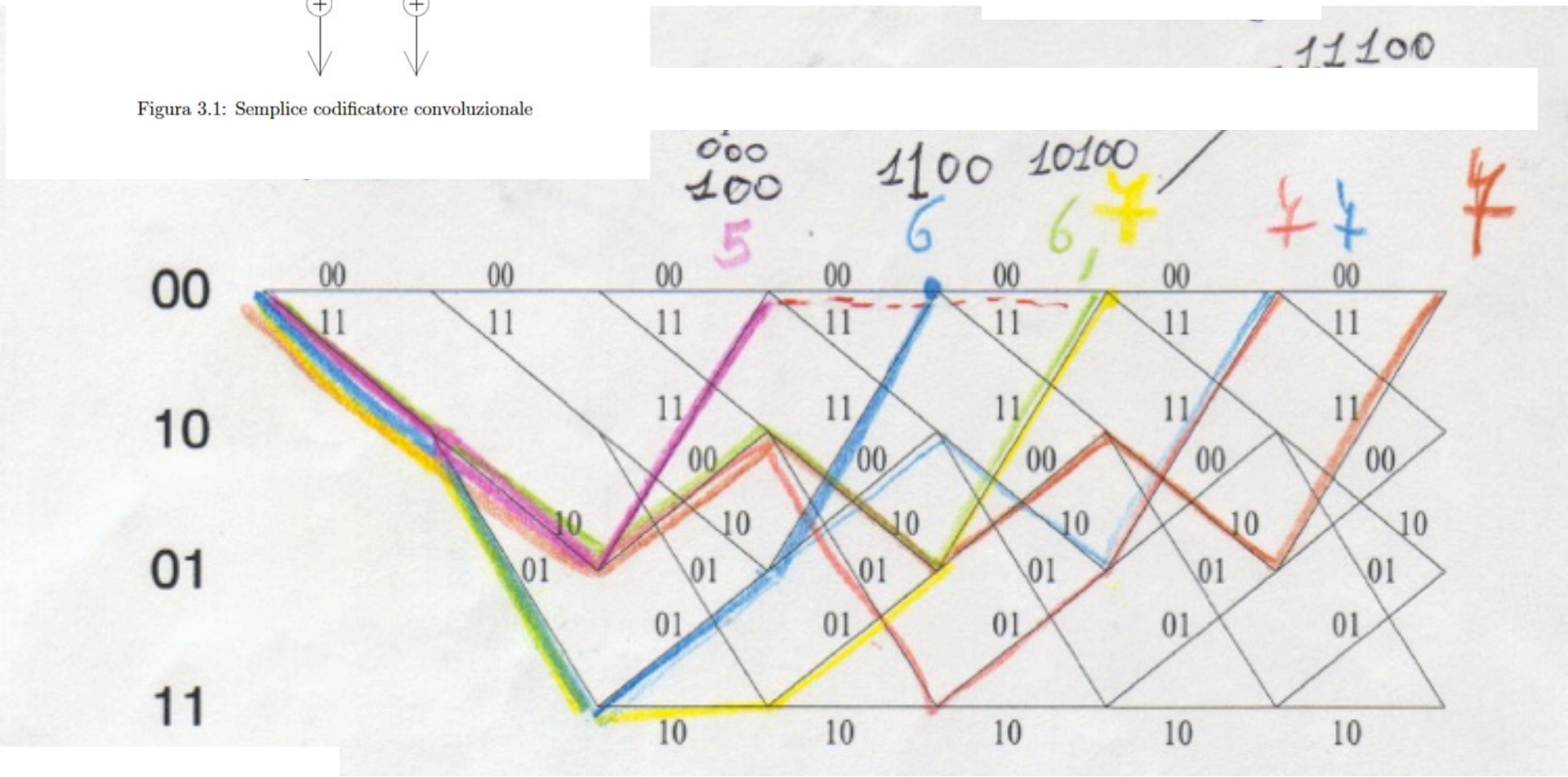
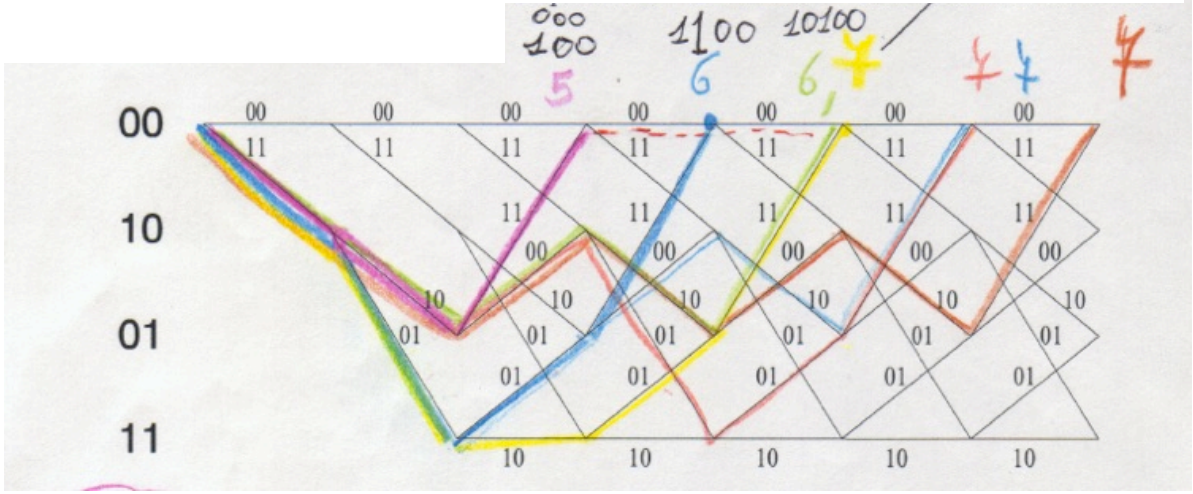


Figura 3.1: Semplice codificatore convoluzionale





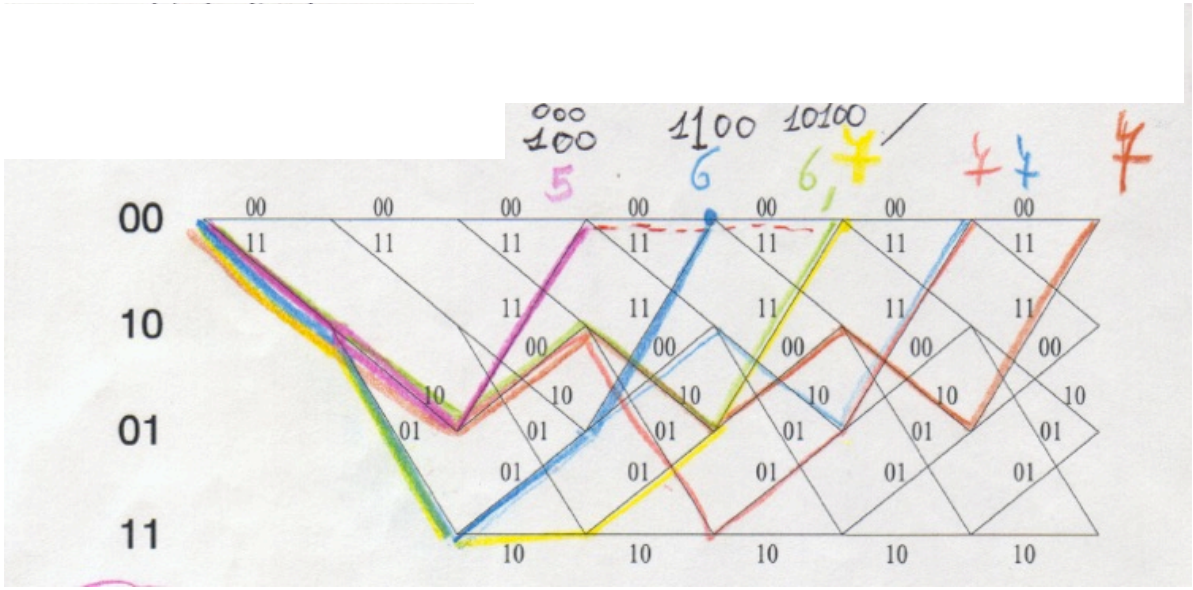
$$P(E) \le Q\left(\sqrt{\frac{2E_b}{N_0}5R}\right) + 2Q\left(\sqrt{\frac{2E_b}{N_0}6R}\right) + 4Q\left(\sqrt{\frac{2E_b}{N_0}7R}\right) + \ldots =$$

$$= \sum_d a(d) Q\left(\sqrt{\frac{2E_b}{N_0}dR}\right) \qquad (5.4)$$

$$G = d_f R$$

a(d) = number of error events at distance d

Pb(E)



$$P_b(E) \le Q\left(\sqrt{\frac{2E_b}{N_0}5R}\right) + 2Q\left(\sqrt{\frac{2E_b}{N_0}6R}\right) + 2Q\left(\sqrt{\frac{2E_b}{N_0}6R}\right) + \ldots =$$

$$= \sum_d \sum_i i\, n(d,i) Q\left(\sqrt{\frac{2E_b}{N_0}dR}\right) = \sum_d w(d) Q\left(\sqrt{\frac{2E_b}{N_0}dR}\right) \qquad (5.6)$$

n(d,i) = number of error events at distance d, with i wrong information bits

w(d) = complessive number of wrong information bits related to error events at distance d

Examples of Efficient Conv. Codes

K	numero di stati	d_f	$w(d_f)$	generatori
3	4	5	1	7,5
4	8	6	2	17,15
5	16	7	4	35,23
6	32	8	2	75,53
7	64	10	36	171,133
8	128	10	2	371,247

Tab. 2 - Codici convoluzionali con *rate* $R = 1/2$

K	numero di stati	d_f	$w(d_f)$	generatori
3	4	8	3	7,7,5
4	8	10	6	17,15,13
5	16	12	12	37,33,25
6	32	13	1	75,53,47
7	64	15	7	171,165,133
8	128	16	1	367,331,225

Tab. 3 - Codici convoluzionali con *rate* $R = 1/3$

Rates different from 1/a (e.g., 2/3, 7/8, …)

$$P_b(E) \leq \frac{1}{b} \sum_d w(d) Q\left(\sqrt{\frac{2E_b}{N_0} dR}\right)$$

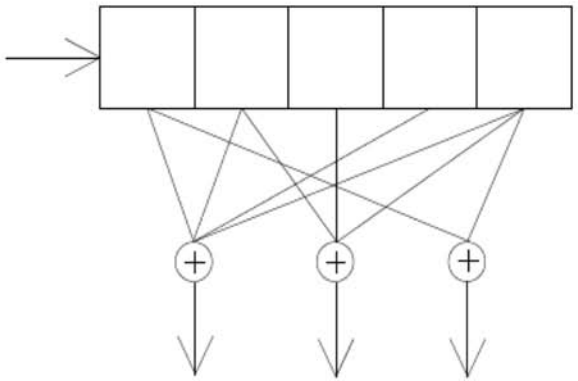


Figura 3.4: Codificatore convoluzionale con *rate* $R = 2/3$ a otto stati; i bit entrano a coppie

Rate 2/3 convolutional encoder, with 8 states.; the input info bits are applied 2 by 2 ..

Rates different from 1/a (e.g., 2/3, 7/8, …)

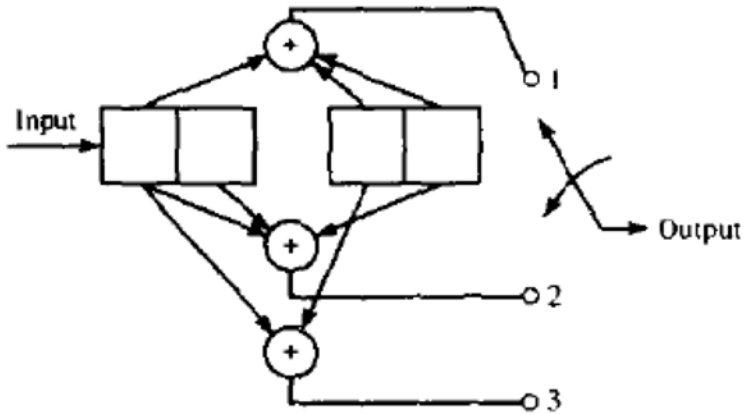


FIGURE 8-2-3 $K = 2,\ k = 2,\ n = 3$ convolutional encoder.

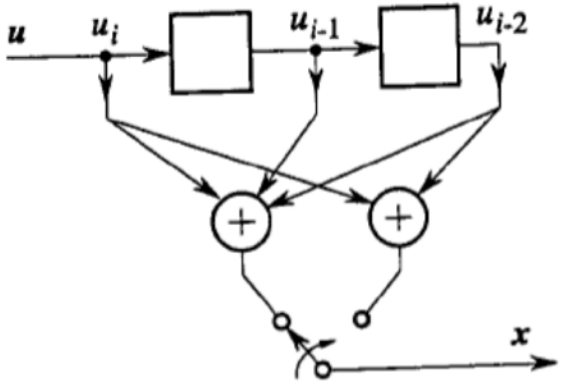
Example 8-2-1

Consider the rate 2/3 convolutional encoder illustrated in Fig. 8-2-3. In this encoder, two bits at a time are shifted into it and three output bits are generated. The generators are

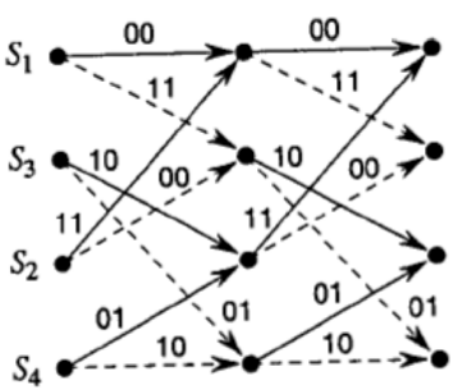
$$\mathbf{g}_1 = [1011], \quad \mathbf{g}_2 = [1101], \quad \mathbf{g}_3 = [1010]$$

In octal form, these generators are (13, 15, 12).

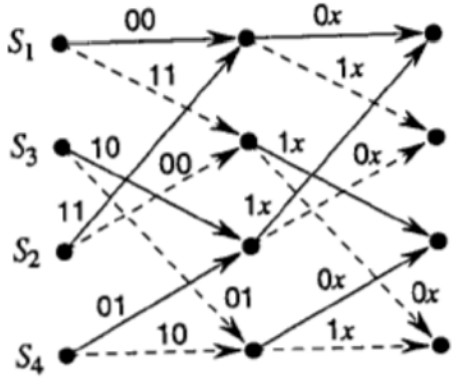
Example 11.6 Consider the 4-state convolutional encoder of Fig. 11.14 (a). For each input bit entering the encoder, two bits are sent through the channel, so that the code generated has rate 1/2. Its trellis is also shown in Fig. 11.14 (b). Suppose now that for every four parity-check digits generated by the encoder, one (the last) is punctured, i.e., not transmitted. In this case, for every two input bits three bits are generated by the encoder, thus producing a rate 2/3 code. The trellis for the punctured code is shown in Fig. 11.14 (c), and the letter "x" denotes a punctured output bit. As an example, the input sequence $\mathbf{u} = 101101\ldots$ would yield $\mathbf{x} = 111000010100$ for the rate 1/2 code, and $\mathbf{x} = 111000010$ for the punctured rate 2/3 code. In a similar way, higher rates can be obtained by increasing the number of punctured parity-check bits.



(a)



(b)



(c)

Figure 11.14: *Encoder (a) and trellis (b) for a (2,1,3) convolutional code. The trellis (c) refers to the rate 2/3 punctured code described in Example 11.6.*

Classic Concatenated Coding (Series)

Concatenated Codes

- Two levels of coding
  Achieves performance of very long code rates while maintaining shorter decoding complexity
- ***Overall rate is*** product of individual code rates
- Codeword error occurs ***if both codes fail***.
- Error probability is found by first evaluating the error probability of "inner" decoder and then evaluating the error probability of "outer" decoder.
- Interleaving is always used with concatenated coding

Concatenated Coding (Series)

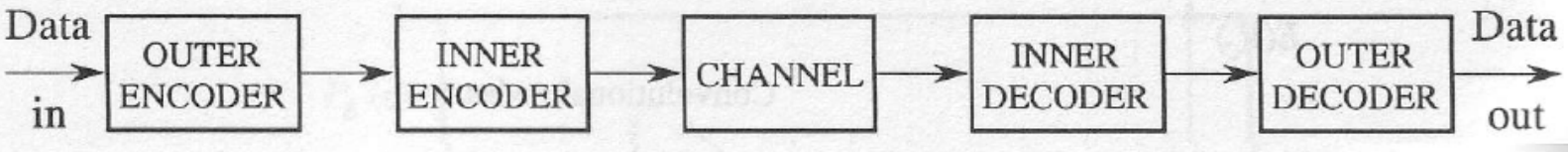
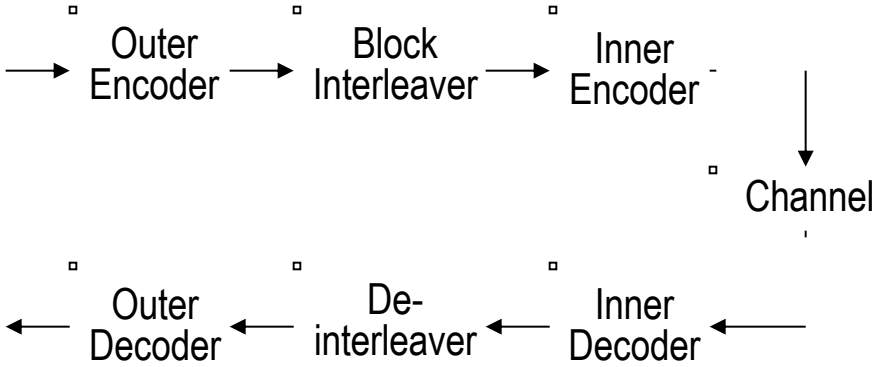


Figure 11.27: *Block diagram of a two-level concatenated code.*



Occorre enumerare i possibili eventi errore, e la distanza geometrica (proporzionale a quella di Hamming) tra il segnale corretto e ciascun concorrente. La linearità del codice consente di assumere che sia stata trasmessa una particolare sequenza, ad esempio quella di tutti zeri. Un evento errore deve terminare nello stato $0 \ldots 0$ e quindi non può avere lunghezza minore di K, mentre non c'è limite alla lunghezza massima. Per codici semplici, come il quattro stati di cui si è visto il traliccio, l'ispezione si può fare manualmente. Ad esempio si individuano in Fig. 3.3 un percorso errato di lunghezza 3 con distanza di Hamming pari a 5, due con distanza pari a 6 e lunghezza rispettivamente 4 e 5, quattro con distanza 7 e lunghezze comprese tra 5 e 7, e così via. Ricordando che $E_s = E_b R$, si ha che la probabilità dell'inizio di un evento errore è maggiorata dallo *union bound*

$$P(E) \leq Q\left(\sqrt{\frac{2E_b}{N_0}5R}\right) + 2Q\left(\sqrt{\frac{2E_b}{N_0}6R}\right) + 4Q\left(\sqrt{\frac{2E_b}{N_0}7R}\right) + \ldots = \tag{3.8}$$
$$= \sum_d a(d) Q\left(\sqrt{\frac{2E_b}{N_0}dR}\right)$$

dove $a(d)$ è il numero di eventi errore a distanza d. Ad alto rapporto segnale-rumore è particolarmente importante la distanza minima del codice. Questa, in generale, non corrisponde all'evento errore di lunghezza minore e va quindi ricercata senza porre limiti a

Se si vuol valutare la probabilità che i bit d'informazione siano errati occorre pesare la probabilità di ogni evento errore con il numero di bit d'informazione errati. Ad esempio si vede che con l'evento errore a distanza minima si sbaglia un solo bit d'informazione (i bit d'informazione decodificati sono 100 anziché 000); con ciascuno dei due a distanza 6 si hanno due bit errati, e così via. La probabilità d'errore sui bit è maggiorata da

$$P_b(E) \leq Q\left(\sqrt{\frac{2E_b}{N_0}5R}\right) + 2Q\left(\sqrt{\frac{2E_b}{N_0}6R}\right) + 2Q\left(\sqrt{\frac{2E_b}{N_0}6R}\right) + \ldots = \\ = \sum_d \sum_i i\, n(d,i) Q\left(\sqrt{\frac{2E_b}{N_0}dR}\right) = \sum_d w(d) Q\left(\sqrt{\frac{2E_b}{N_0}dR}\right) \tag{3.10}$$

dove $n(d,i)$ è il numero di eventi errore a distanza d e con i bit d'informazione errati, e $w(d) = \sum i\, n(d,i)$ è il numero complessivo di bit d'informazione errati negli eventi errori aventi distanza d. La formula tiene implicitamente conto non solo degli eventi errore che iniziano ad un generico istante, ma anche di quelli già in corso. Infatti la frequenza con cui *inizia* un evento errore viene moltiplicata per il numero *complessivo* di bit d'informazione errati che esso produce, e non solo per quello prodotto nella prima transizione di stato. Se il numeratore b del *rate* del codice è diverso da uno occorre anche tener conto del numero di bit trasmessi per ciascuna transizione di stato e si ha

$$P_b(E) \leq \frac{1}{b} \sum_d w(d) Q\left(\sqrt{\frac{2E_b}{N_0}dR}\right) \tag{3.11}$$