3F4: Data Transmission

Handout 11: Decoding Convolutional Codes

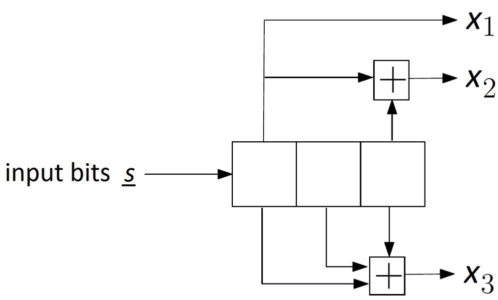
Ioannis Kontoyiannis
[based on notes by Ramji Venkataramanan]

Signal Processing and Communications Lab
Department of Engineering
i.kontoyiannis@eng.cam.ac.uk

Lent Term 2019

Convolutional codes

In convolutional codes, a stream of input bits is transformed into a stream of code bits using a shift register (filter)



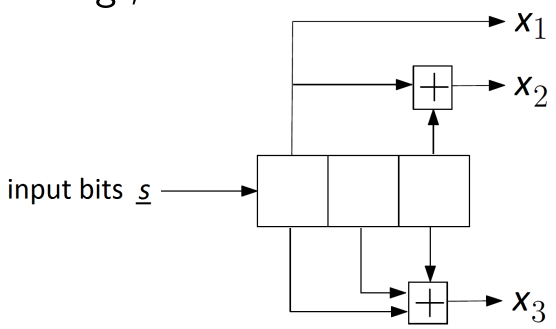
- Here, for every $k = 1$ data bit, we have $n = 3$ code bits
- Assuming the initial state of the shift register is (0 0 0) the code bits corresponding to the input $\underline{s} = 1010$ are

$$(111\ 001\ 100\ 001)$$

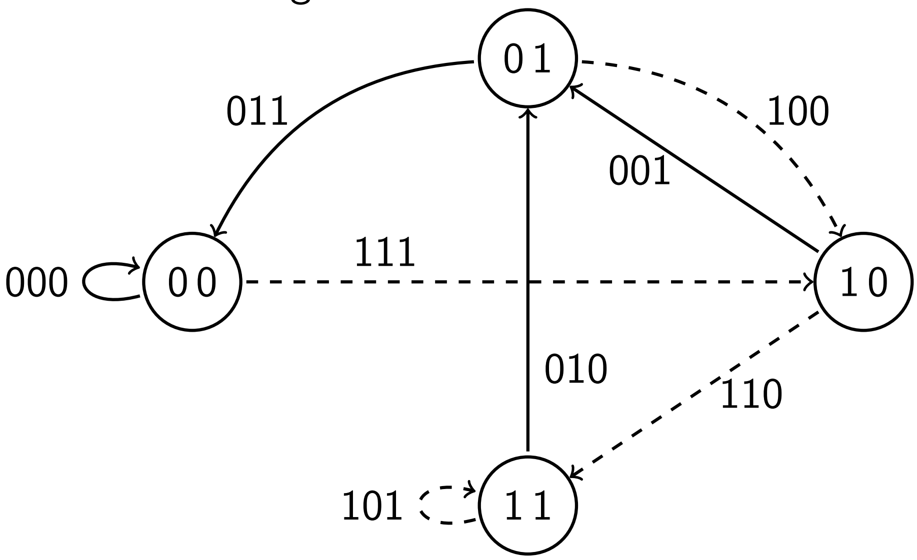
- Dependence between code bits is created via the shift register

Finite-state machine description

Convolutional codes can also be represented via *state diagrams* corresponding to *finite-state machines*. E.g., the code:

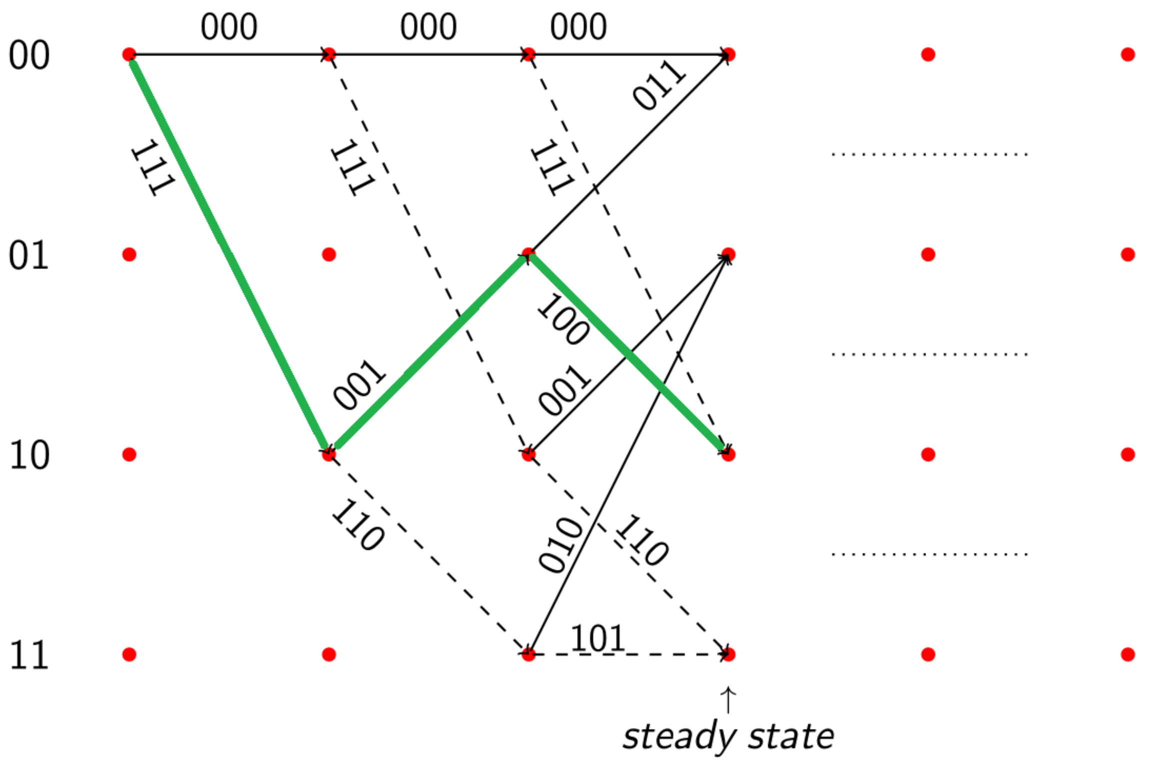


has the state diagram:



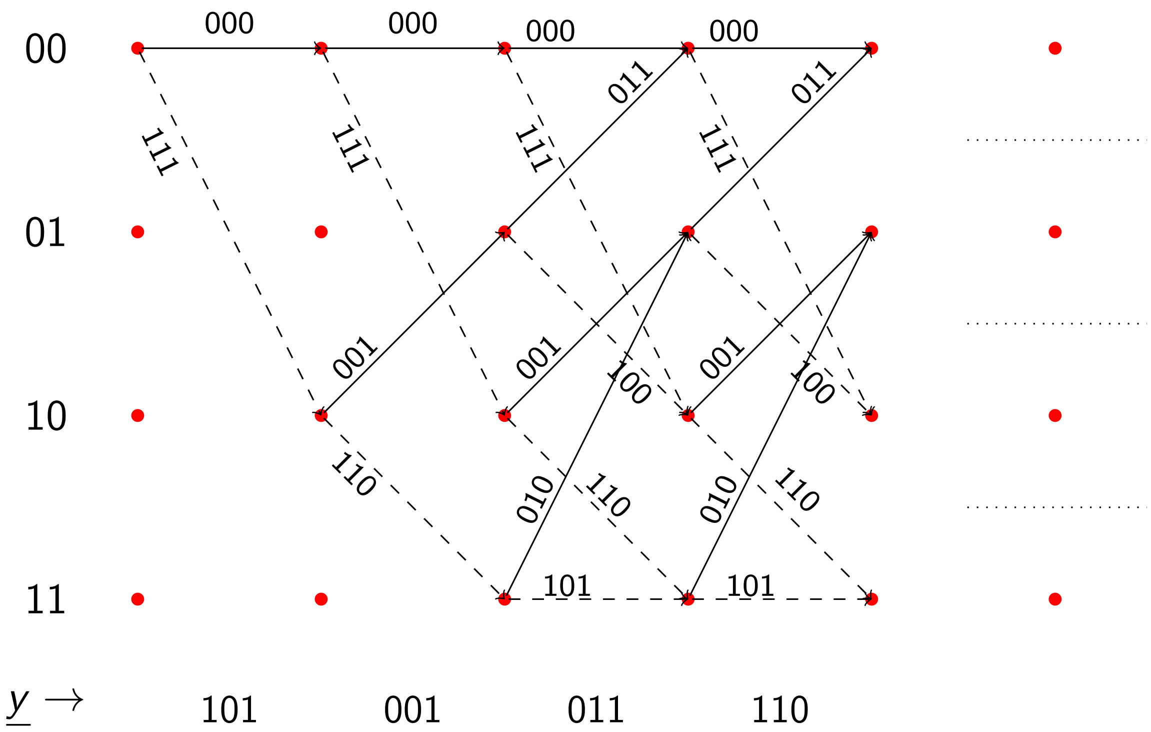
The trellis representation

E.g.: The path traced by the input string $\underline{s} = 101$ producing the code string $\underline{x} = 111\ 001\ 100$



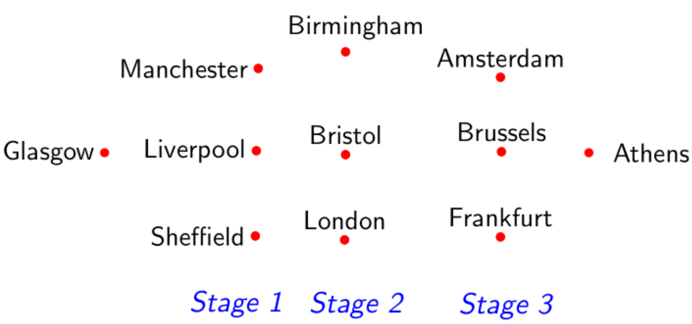
Decoding convolutional codes

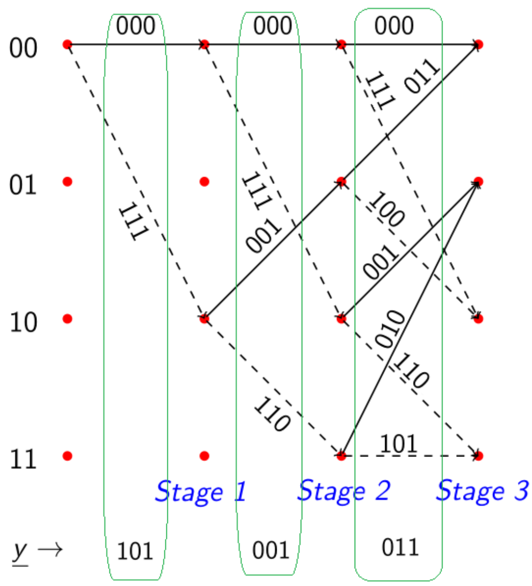
Recall: Optimal decoding is minimum distance decoding
E.g.: If we receive $\underline{y} = 101\ 001\ 011\ 111$, we need to find a path in the trellis which gives a code sequence $\hat{\underline{x}}$ minimising $d(\underline{y}, \hat{\underline{x}})$



The Viterbi algorithm

The Viterbi algorithm is simply dynamic programming
on the trellis corresponding to a convolutional code
Except easier: Not all paths are allowed!

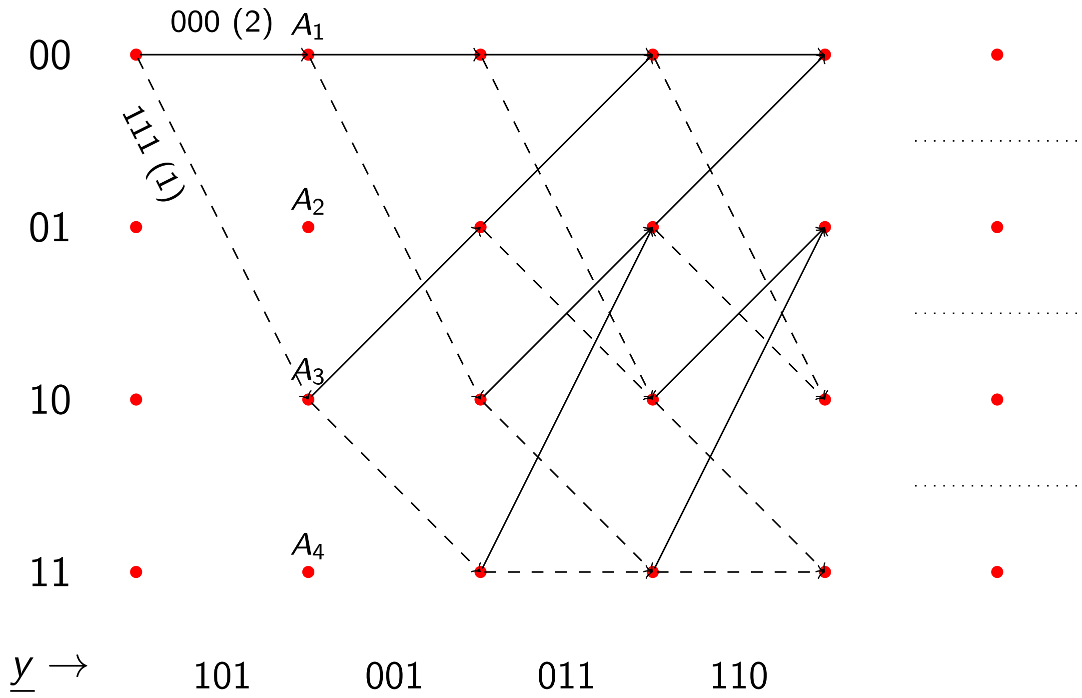




cities $\mapsto$ states
distance $\mapsto$ Hamming dist between produced and received code bits

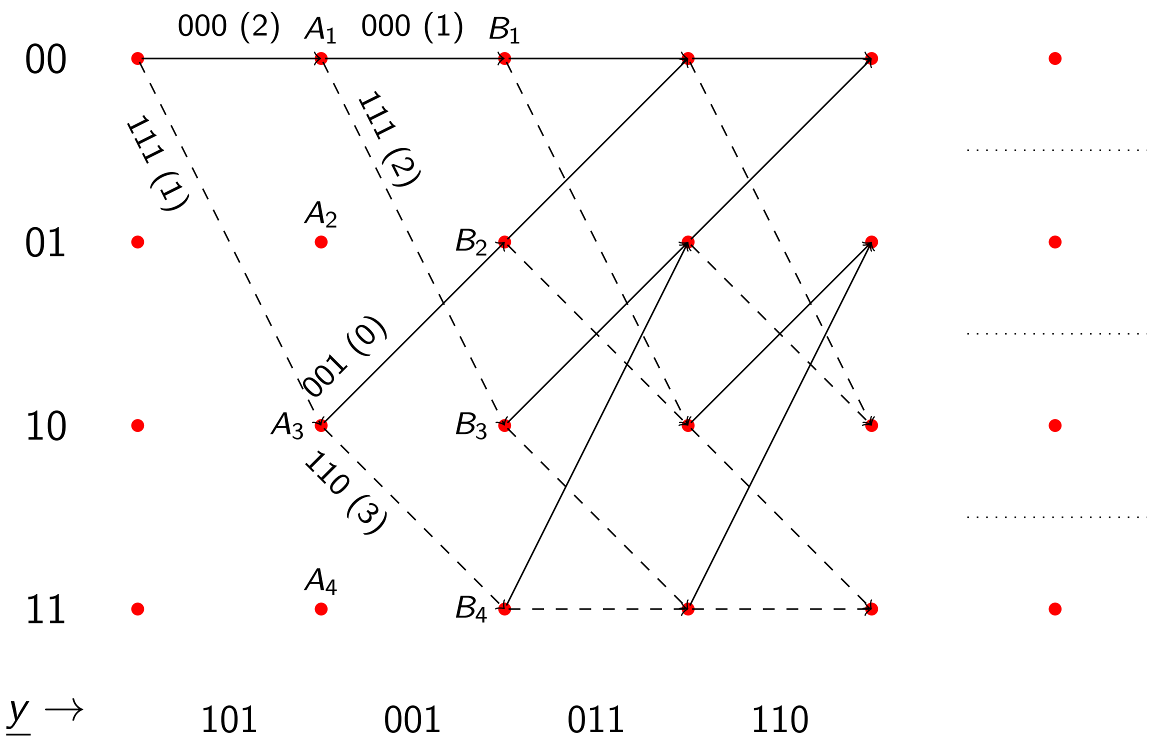
The Viterbi algorithm

E.g., as before: To decode $\underline{y} = 101\ 001\ 011\ 111$, need to find the path in the trellis corresponding to that $\underline{\hat{x}}$ which minimizes $d(\underline{y}, \underline{\hat{x}})$. First stage:



Observe: The number in parentheses is the *Hamming distance* between received and produced code bits

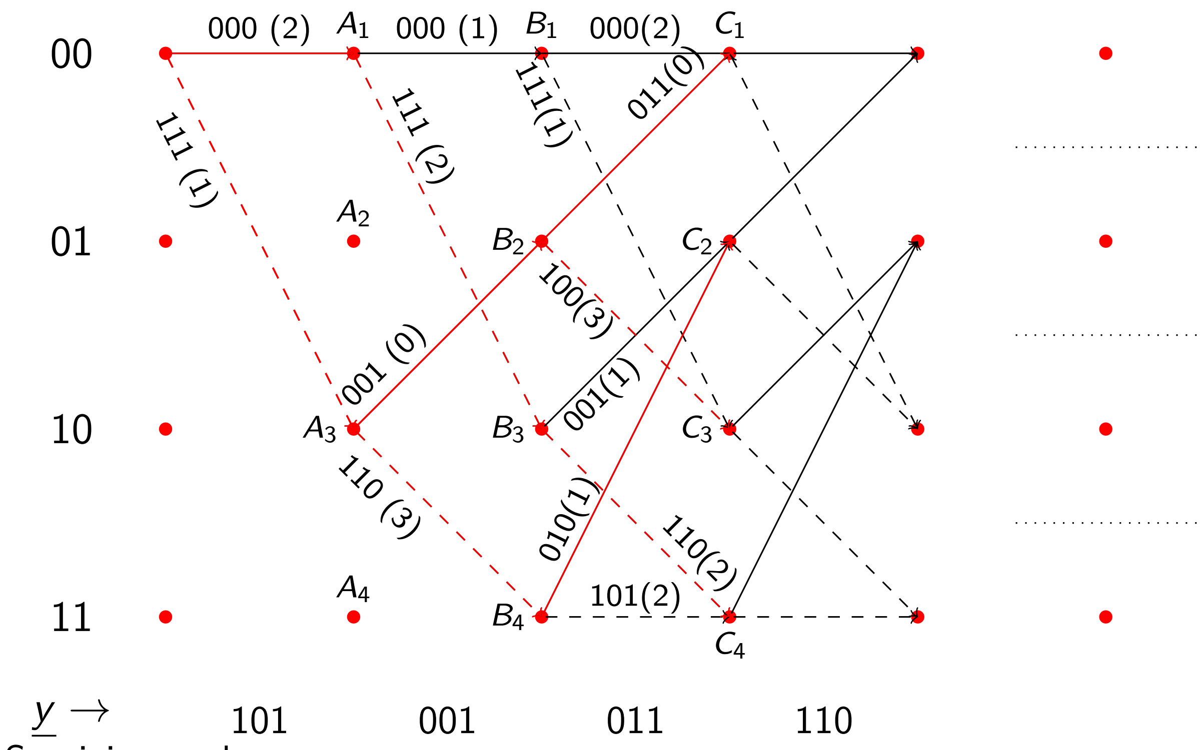
Second stage:



We only have one path from 00 to each of B_1, B_2, B_3, B_4.
The distances from the corresponding received bits are

$$d(00, B_1) = 3,\ d(00, B_2) = 1,\ d(00, B_3) = 4,\ d(00, B_4) = 4$$

Third stage:



Surviving paths:

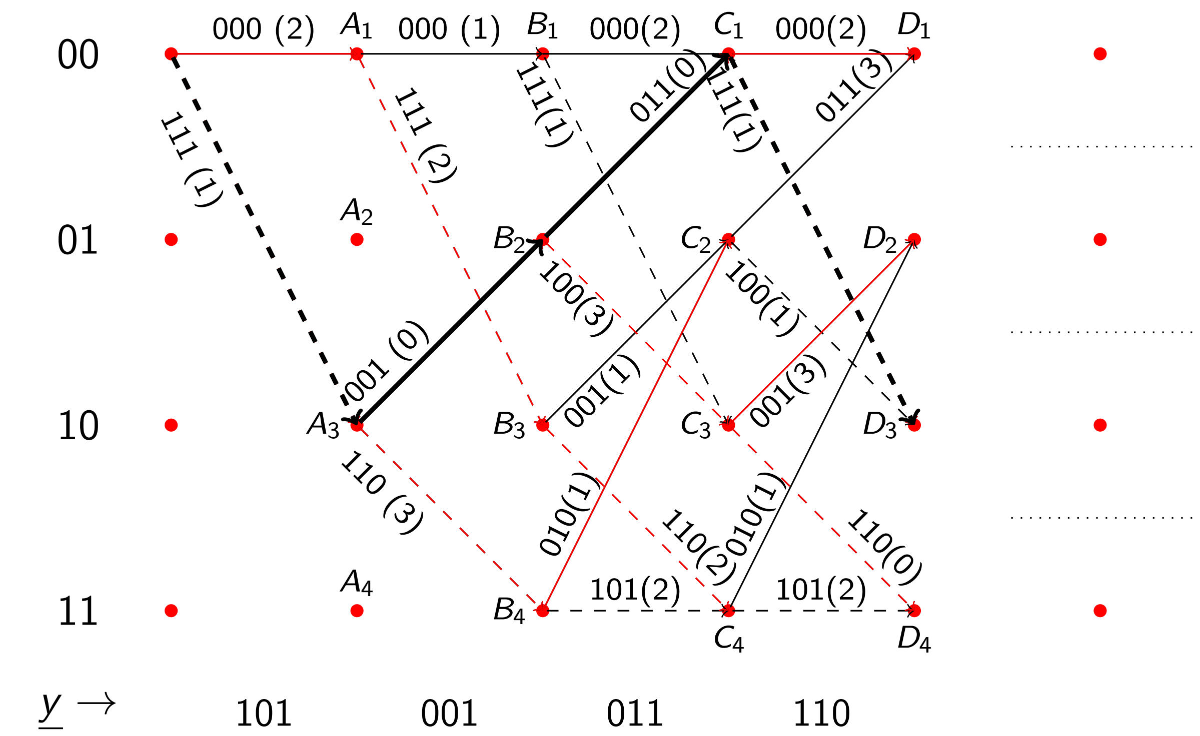
$d(00, C_1) = \min\{d(00, B_1)+2, d(00, B_2)+0\} = 1; Path : 00-A_3-B_2-C_1$

$d(00, C_2) = \min\{d(00, B_3)+1, d(00, B_4)+1\} = 5; Path : 00-A_3-B_4-C_2$

$d(00, C_3) = \min\{d(00, B_2)+3, d(00, B_1)+1\} = 4; Path : 00-A_3-B_2-C_3$

$d(00, C_4) = \min\{d(00, B_3)+2, d(00, B_4)+2\} = 6; Path : 00-A_1-B_3-C_4$

Fourth stage:



Surviving paths:

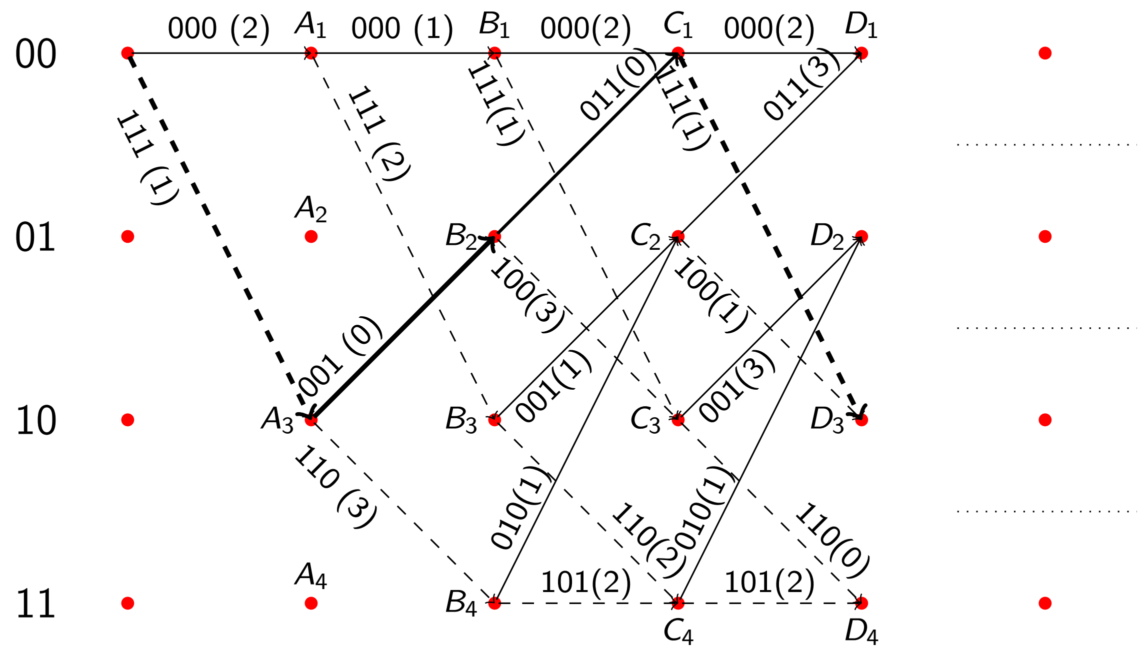
$d(00, D_1) = \min\{d(00, C_1)+2, d(00, C_2)+3\} = 3;\ 00-A_3-B_2-C_1-D_1$

$d(00, D_2) = \min\{d(00, C_3)+3, d(00, C_4)+1\} = 7;\ 00-A_3-B_2-C_3-D_2$

$d(00, D_3) = \min\{d(00, C_2)+1, d(00, C_1)+1\} = 2;\ 00-A_3-B_2-C_1-D_3$

$d(00, D_4) = \min\{d(00, C_3)+0, d(00, C_4)+2\} = 4;\ 00-A_3-B_2-C_3-D_4$

Finally: Decoded codeword $\hat{x}$:



$\underline{y} \rightarrow$ 101 001 011 110

The min-distance path is shown in bold, corresponding to:

$$\hat{x} = 111\ 001\ 011\ 111$$

And the corresponding source sequence, easily determined from the state transitions, is: $\hat{s} = 1001$

Aside: Yet another representation of a convolutional code

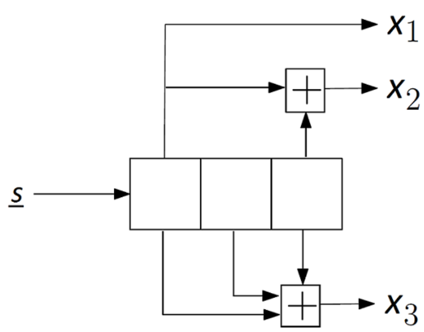
Input sequence: $\underline{s} = (s_1, \ldots, s_m)$

Generators: $g_1 = (1\,0\,0), \quad g_2 = (1\,0\,1), \quad g_3 = (1\,1\,1)$

Code bits: $\underline{x} = (x_1^{(1)}, x_2^{(1)}, x_3^{(1)}, x_1^{(2)}, x_2^{(2)}, x_3^{(2)}, \ldots, x_1^{(m)}, x_2^{(m)}, x_3^{(m)})$

Observe: We obtain $\underline{x}$ simply by interleaving the convolutions of $\underline{s}$ with each of the generators: $\underline{x}_1 = \underline{s} * g_1$

$$\underline{x}_2 = \underline{s} * g_2$$
$$\underline{x}_3 = \underline{s} * g_3$$



Alternatively: We can express this in terms of the Z-transform (with $Z = z^{-1}$):

$$g_1(Z) = 1$$
$$g_2(Z) = 1 + Z^2$$
$$g_3(Z) = 1 + Z + Z^2$$

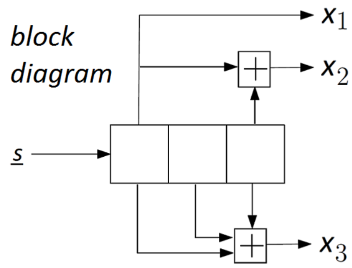
so that: $\quad x(Z) = s(Z^3)g_1(Z^3) + Zs(Z^3)g_2(Z^3) + Z^2 s(Z^3)g_3(Z^3)$

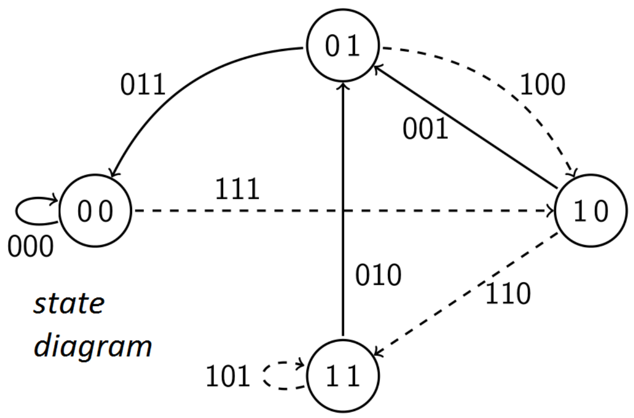
Performance of convolutional codes

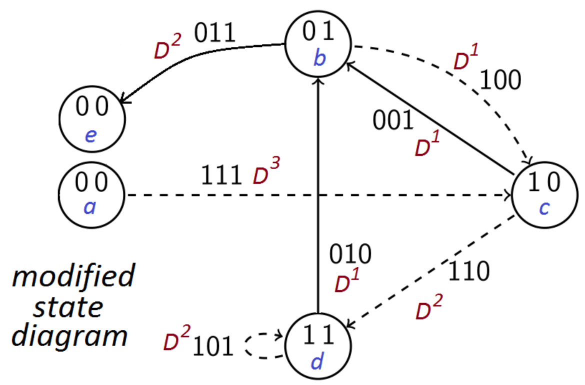
- Since convolutional codes are linear codes
  d_{min} = the minimum weight among nonzero codewords
  = minimum distance between any codeword $\underline{x}$ and $\underline{0}$
- But since convolutional codes do not have a fixed blocklength we do not examine their minimum distance d_{min}
- Instead we will consider the **free distance d_{free}**
  defined as *the min weight among all codewords generated with the code starting and ending in the all-zero state*
- *Interpretation.* If $d_{free} = d$, then any collection of $\leq \lfloor \frac{d-1}{2} \rfloor$ errors can be corrected, as long as these "error bursts" are not "too close" to one another
- Simplest to compute d_{free} via the code's **transfer function**
- To define the transfer function, first modify the state diagram:
  (i) simplify state labels
  (ii) duplicate the all-zero state (as start and end)
  (iii) label each transition by D raised to the
  Hamming weight of the sequence it produces

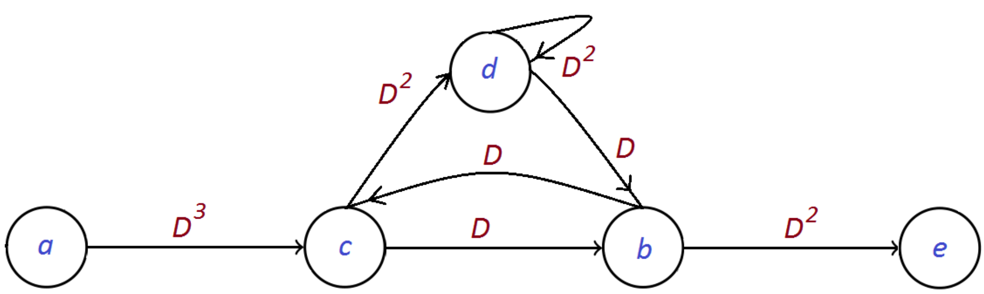
Modified state diagram

(i) simplify state labels, (ii) duplicate all-zero state
(iii) label each transition by D raised to the
Hamming weight of the sequence it produces





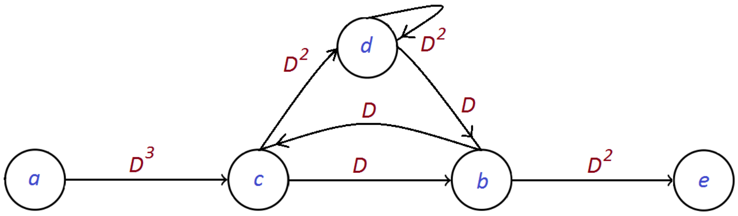




State equations and transfer function

From the diagram we obtain the **state equations**

$X_b = DX_c + DX_d$
$X_c = D^3X_a + DX_b$
$X_d = D^2X_c + DX_d$
$X_e = D^2X_b$



We define the **transfer function**: $T(D) = X_e/X_a$

Solving this system:

$$T(D) = D^6/(1-2D^2) = D^6 + 2D^8 + 4D^{10} + 8D^{12} + \cdots$$

Interpretation

The first term in $T(D) \Rightarrow$ there is a single path of weight $d = 6$ starting and ending in state 00; from the state diagram we see it is *acbe*

Second term $\Rightarrow$ there are exactly two paths from 00 to 00 of weight $d = 8$: *acdbe* and *acbcbe*

Third term $\Rightarrow$ there are four paths of weight $d = 10$, etc.

- In particular, $\boldsymbol{d_{free} = 6}$

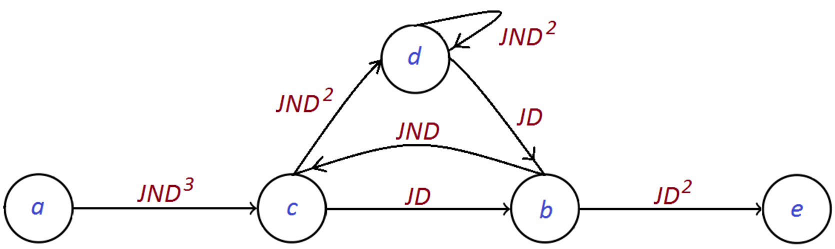
The extended transfer function

In fact, it is easy to similarly get more information than only on the Hamming weight of paths

Extend the state diagram further:

Add a "J" to every branch, counting total path length

Add an "N" to each branch corresponding to an input bit "1"



Obtain from the diagram the **extended state equations**

$$\begin{aligned} X_b &= JDX_c + JDX_d \\ X_c &= JND^3X_a + JNDX_b \\ X_d &= JND^2X_c + JND^2X_d \\ X_e &= JD^2X_b \end{aligned}$$

Define the **extended transfer function** $T(J, N, D) = X_e/X_a$

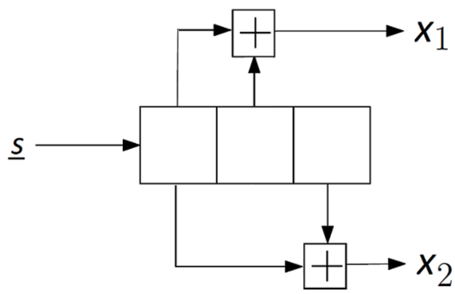
Extended transfer function: Interpretation

Again, solving the above system:

$$\begin{aligned} T(J,N,D) &= \frac{J^3ND^6}{1 - JND^2(1+J)} \\ &= J^3ND^6 + [J^4N^2D^8 + J^5N^2D^8] \\ &\quad +[J^5N^3D^{10} + 2J^6N^3D^{10} + J^7N^3D^{10}] + \cdots \end{aligned}$$

- First term: There is exactly one path of Hamming weight $d = 6$, it has length 3, & of its 3 source bits exactly one is a 1
- Second and third terms: There are two paths of Hamming weight $d = 8$, one of length 4, and one of length 5, and the information sequence of each path contains two 1s
- Note: If we are transmitting a length-m source sequence $\underline{s} = (s_1, \ldots, s_m)$, we can truncate the expansion of the extended transfer function after all the terms of order J^m

Warning: Catastrophic encoders



Consider the encoder shown, with input
$\underline{s} = 111111111 \cdots$
The code sequence generated is
$\underline{x} = 11\ 01\ 00\ 00\ 00\ 00\ 00\ 00\ 00\ 00 \cdots$
Suppose the received sequence is
$\underline{y} = 11\ 10\ 01\ 00\ 00\ 00\ 00\ 00\ 00\ 00 \cdots$
This is itself a codeword, corresponding to source string
$\underline{s}' = 100000 \cdots$

$\Rightarrow$ A finite number of channel errors (only three, in fact) led to infinitely many decoding errors!

- Encoders with this very undesirable property are called *catastrophic*
  - The root of the problem is that we can go from state 11 to state 11, with an input bit of weight 1 and an output string 00 of weight 0

Avoiding catastrophic encoders

Suppose an encoder of an S-stage shift register
has generators $\{g_1, g_2, \ldots, g_n\}$
with corresponding Z-transforms (again with $Z = z^1$):
$\{g_1(Z), g_2(Z), \ldots, g_n(Z)\}$

Theorem [Massey & Sain, 1968]

If the greatest common divisor of the polynomials
$\{g_1(Z), g_2(Z), \ldots, g_n(Z)\}$ is of the form Z^ℓ
for some integer $\ell \geq 0$, then the encoder defined by the generators
$\{g_1, g_2, \ldots, g_n\}$ is *not* catastrophic

Exercise. Show that the encoder of the previous slide
does not satify the conditions of the theorem

Summary: Convolutional codes

- Information bits are fed into an S-stage shift register
- Code bits are linear combinations of the contents of the register
- The code can be represented by a block diagram, a state diagram, or a trellis diagram
- In each representation there are 2^{S-1} states, each state representing the bits in the first $S-1$ cells of the register
- Each state transition caused by an input bit is labeled with a sequence of code bits
- Efficient minimum distance decoding via Viterbi algorithm: Complexity scales *linearly* with number of input bits
- Code properties are summarized in the transfer function
- Error correcting ability described by the free distance instead of the minimum distance of linear codes

Convolutional codes: Applications and extensions

- Convolutional codes are very widely used, e.g., in digital video, wireless communications including today's popular wireless standards (such as 802.11), and satellite links
- *Turbo codes,* in which two convolutional codes cleverly interact with one another, are the state-of-the-art in mobile communication systems
- The same ideas easily generalize to non-binary codes

* The exact same process of Viterbi decoding generalizes for transmission over Gaussian noise channels:
Map x's as $1 \mapsto +1$ and $0 \mapsto -1$, so that the channel output is $Y_i = x_i + Z_i$, where $Z_i \sim N(0, N)$ is independent noise.
Then maximum likelihood decoding becomes minimum distance decoding wrt Euclidean instead of Hamming distance

Ex. In this case, repeat Exercise 3 ii) in Examples Paper 3, with $\underline{Y} = (0.2, -1, 1.9, 1.35, -0.1, 0.4, -1.7, 0.6, 0.8, 3, 2.1, -0.4)$.