ELG 5372 Error Control Coding

Lecture 22: Soft Decision Decoding of Convolutional Codes.

uOttawa

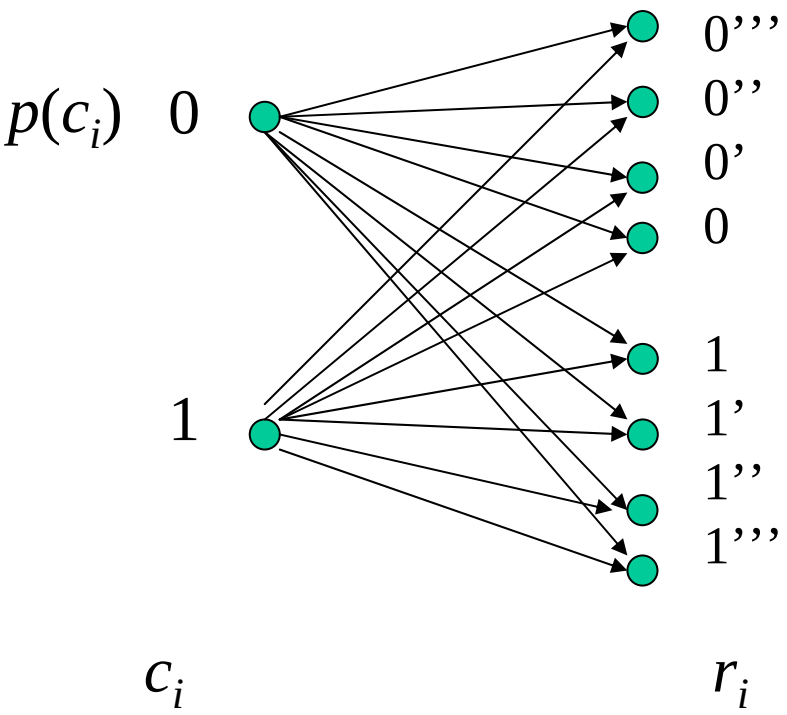
L'Université canadienne
Canada's university

Université d'Ottawa | University of Ottawa

uOttawa.ca

Likelihoods

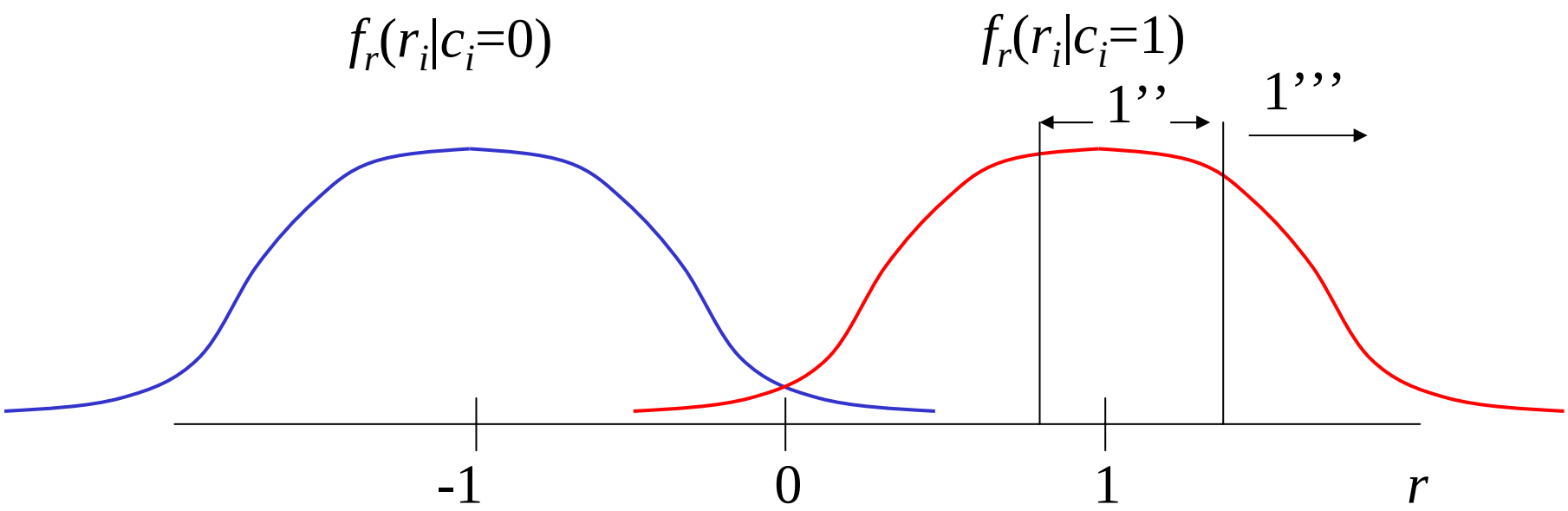
- Consider the following channel



$p(r_i|c_i)$ are known

likelihood

AWGN Channel and Likelihoods



Maximum Likelihood Sequence Estimation

- Suppose we receive **r** = (r_0, r_1, r_2) = (1'', 0'', 0).
- In the hard decision case, this would be given to the decoder as (1,0,0).
- In the coded case, suppose that the only possible code sequences are (0, 0, 0), (0,1,1), (1,0,1) and (1,1,0).
- Assuming that 0 and 1 are transmitted with equal probability, the most likely codeword is the one that maximizes the following:

$$\max_{\mathbf{c}} \prod_{i=1}^{3} p(r_i \mid c_i)$$

Example

c_i \ r_i	0’’’	0’’	0’	0	1	1’	1’’	1’’’
0	0.47	0.25	0.139	0.085	0.0455	0.009	0.001	0.0005
1	0.0005	0.001	0.009	0.0455	0.085	0.139	0.25	0.47

Assume that 0 and 1 are transmitted with equal probability.

Example

- Then $p(\mathbf{r}|000) = 0.001 \times 0.25 \times 0.085 = 0.000021$.
- $p(\mathbf{r}|010) = 0.001 \times 0.001 \times 0.085 = 8.5 \times 10^{-9}$.
- $p(\mathbf{r}|101) = 0.25 \times 0.25 \times 0.0455 = 0.0028$.
- $p(\mathbf{r}|110) = 0.25 \times 0.001 \times 0.085 = 0.000021$.
- In hard decision case, the decoder would have determined that 000, 101 and 110 are all equally likely.

Log likelihood function

- The likelihood function is a product of conditional probabilities.
- For long sequences, the resulting likelihoods will be small compared to 1.
- To simplify, we use the log likelihood function.
- $p(r_i|c_i)$ is expressed as $\log(p(r_i|c_i))$ and

$$\prod_{i=1}^{L} p(r_i \mid c_i) \text{ becomes } \sum_{i=1}^{L} \log(p(r_i|c_i)) \qquad (1)$$

The most likely codeword maximizes (1)

Decoding Metrics

- $M(r_i|c_i)$ is a function of the log likelihood function.
- Since the log of a probability is always negative, we will add a constant to all log likelihood functions so that they are all positive.
- Log likelihood functions may also have many digits after the decimal, so we multiply by another constant to yield metrics that can be approximated by whole numbers (or numbers that don't require much memory).
- $M(r_i|c_i) = a(\log(p(r_i|c_i)+b)$.
- The path metric is

$$\sum_{i=1}^{L} M(r_i \mid c_i) = \sum_{i=1}^{L} a(\log(p(r_i \mid c_i) + b) = a\sum_{i=1}^{L} \log(p(r_i \mid c_i)) + abL$$

r_i / c_i	0’’’	0’’	0’	0	1	1’	1’’	1’’’
0	0.47	0.25	0.139	0.085	0.0455	0.009	0.001	0.0005
1	0.0005	0.001	0.009	0.0455	0.085	0.139	0.25	0.47

r_i / c_i	0’’’	0’’	0’	0	1	1’	1’’	1’’’
0	-0.328	-0.612	-0.857	-1.07	-1.34	-2.05	-3	-3.3
1	-3.3	-3	-2.05	-1.34	-1.07	-0.857	-0.612	-0.328

r_i / c_i	0’’’	0’’	0’	0	1	1’	1’’	1’’’
0	2.972	2.688	2.443	2.23	1.96	1.25	0.3	0
1	0	0.3	1.25	1.96	2.23	2.443	2.688	2.972

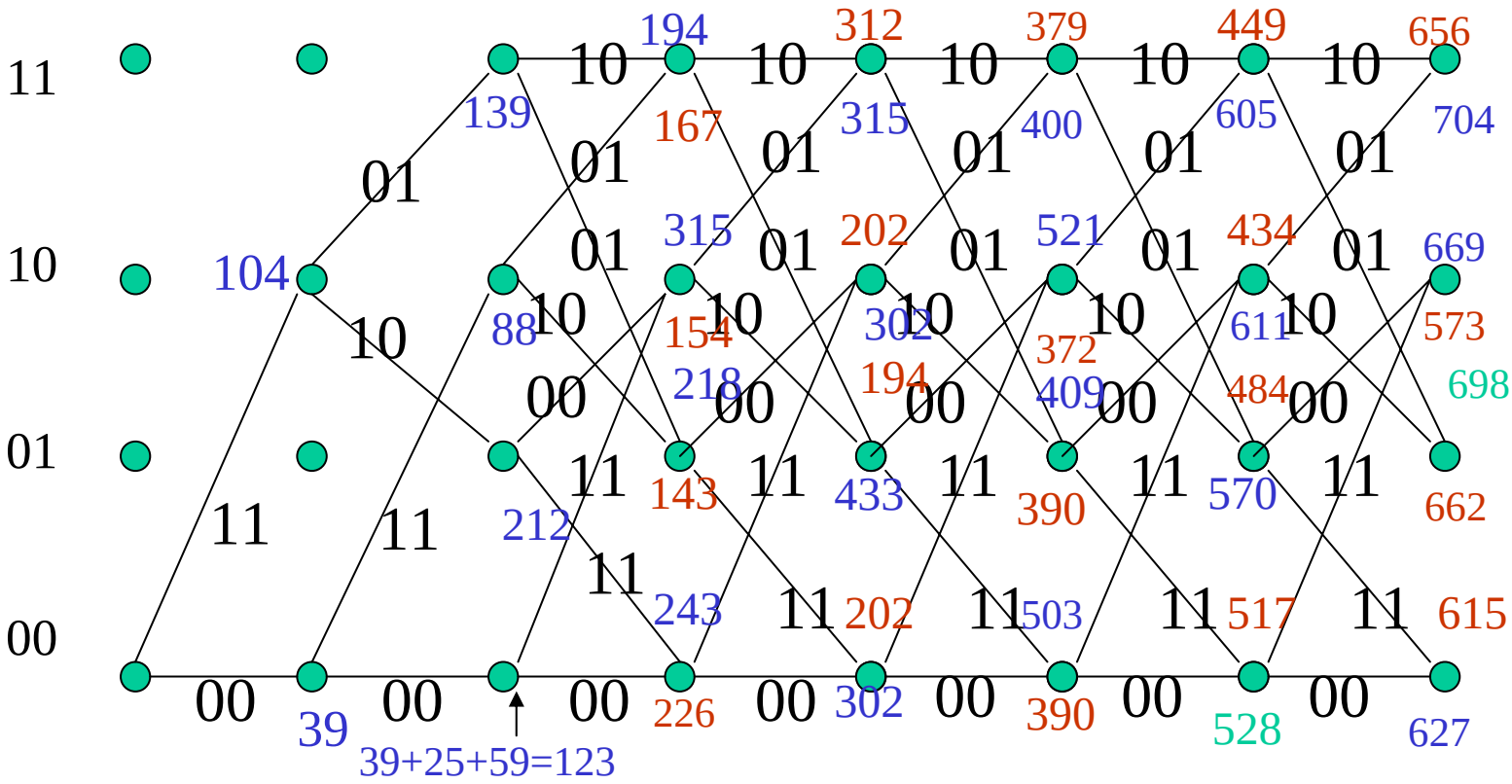
Add 3.3 to all then multiply by a constant an round (I used 20).

r_i / c_i	0’’’	0’’	0’	0	1	1’	1’’	1’’’
0	59	54	49	45	39	25	6	0
1	0	6	25	39	45	49	54	59

Example

- Again consider **r** = (1’’, 0’’, 0).
- $M(\mathbf{r}|000) = 6+54+45 = 105$
- $M(\mathbf{r}|011) = 6+6+39 = 51$
- $M(\mathbf{r}|101) = 54+54+39 = 147$
- $M(\mathbf{r}|110) = 54+6+45 = 105$

Metrics applied to Viterbi Algorithm



r = 1’’’1, 1’0’’’, 0’’0’, 1’’’0’’’, 0’1, 1’1’’’, 0’’0

Metrics applied to Viterbi Algorithm

- If the encoder is reset to all zero state, then the decoded stream must end in all zero state. Therefore the decoded path would be
  - 00-10-01-10-01-10-01-00 (state transitions)
  - 11, 10, 00, 10, 00, 10, 11 (code sequence)
  - 1,0,1,0,1,0,0 (message)

Soft Decision Decoding

- In the previous example, we used quantized values for the received sequence.
- This is not pure soft decision decoding, but rather a compromise between soft decision and hard decision decoding.
- In soft decision decoding, infinite quantization is used (in other words, we can use the decision variable output or perhaps a log likelihood ratio).
- The code sequence that maximizes $\prod_{i=1}^{L} p(r_i \mid c_i)$ is also the code sequence that has the smallest Euclidean distance from **r**.

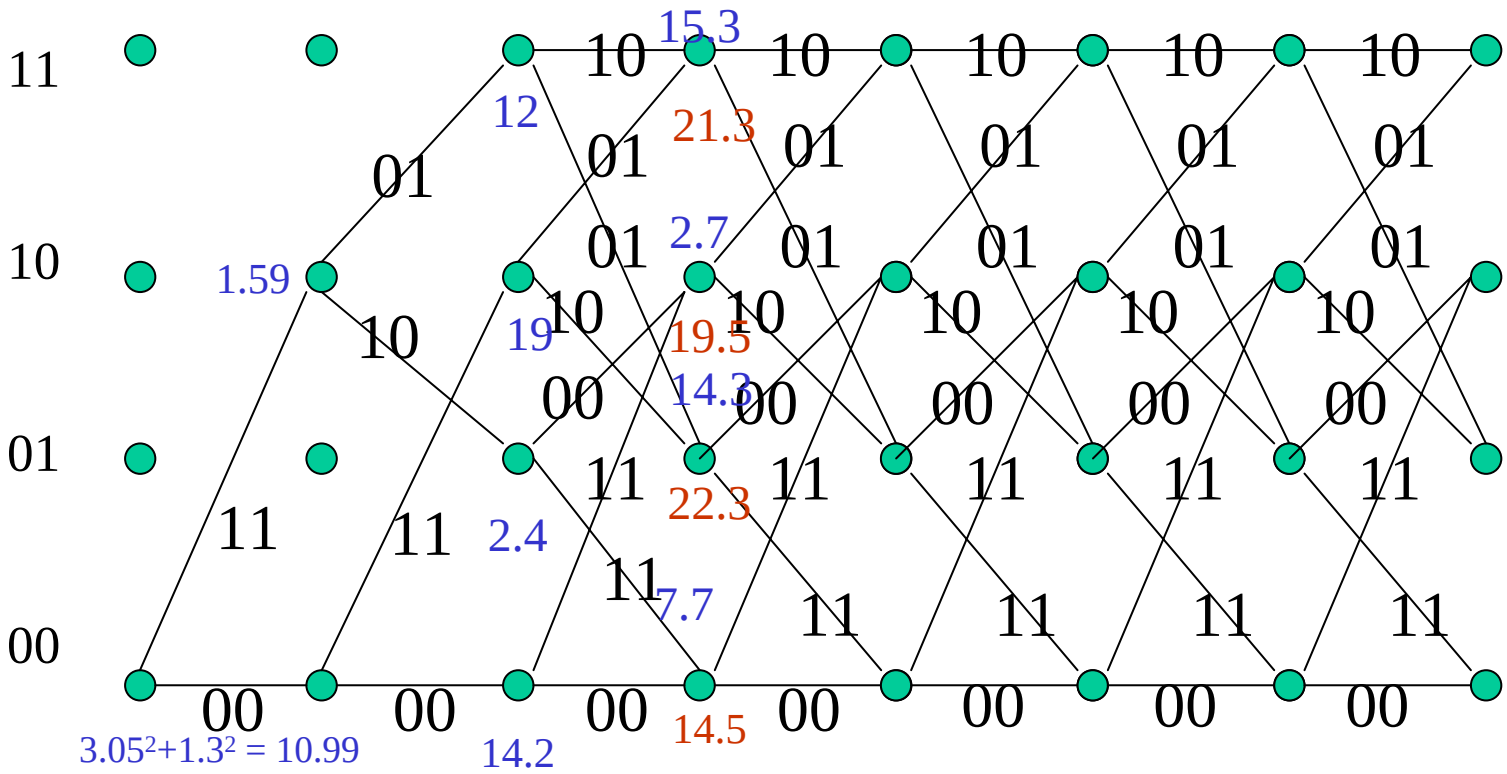
Euclidean Distance

- Let **v** = (v_1, v_2, v_3, … v_L) and let **u** = (u_1, u_2, …, u_L).
- The Euclidean distance between **v** and **u** is:

$$ED(\mathbf{v},\mathbf{u}) = \sqrt{\sum_{i=1}^{L}(v_i - u_i)^2}$$

- If **c** minimizes *ED*(**r**,**c**), then **c** also minimizes ED^2(**r**,**c**) (since distances cannot be negative).

Example

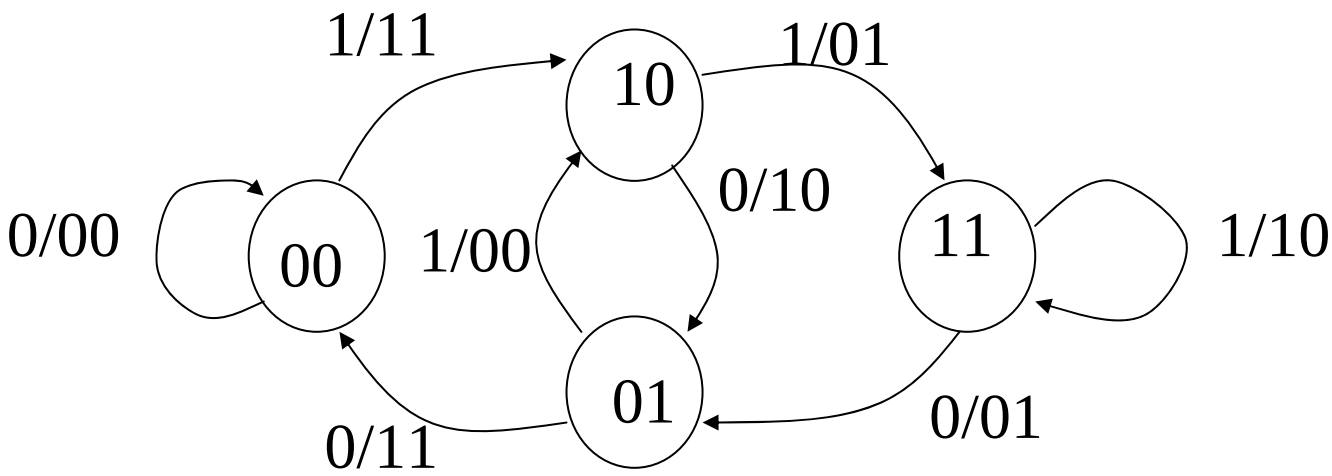


r = 2.05 0.3, 0.6 -1.8, -0.75 -0.5, 1.44 -1.3, -0.5 0.21, 0.55 1.8, -0.8 -0.2

Assuming a 1 is received as 1 and a 0 as -1 in the absence of noise.

Code Transfer Function

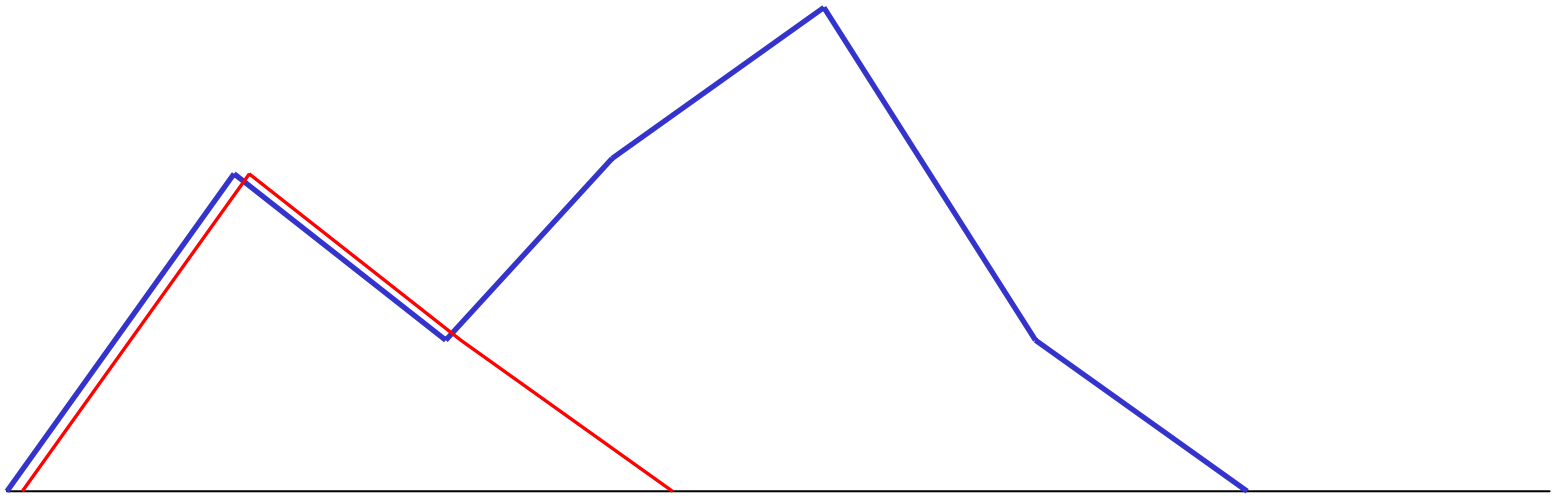
Consider the rate ½ code $\mathbf{G}(D) = [1+D+D^2, 1+D]$. The state diagram is:



The code transfer function tells us how many paths there are of weight d as well as the weight of the message sequences that produce this path.

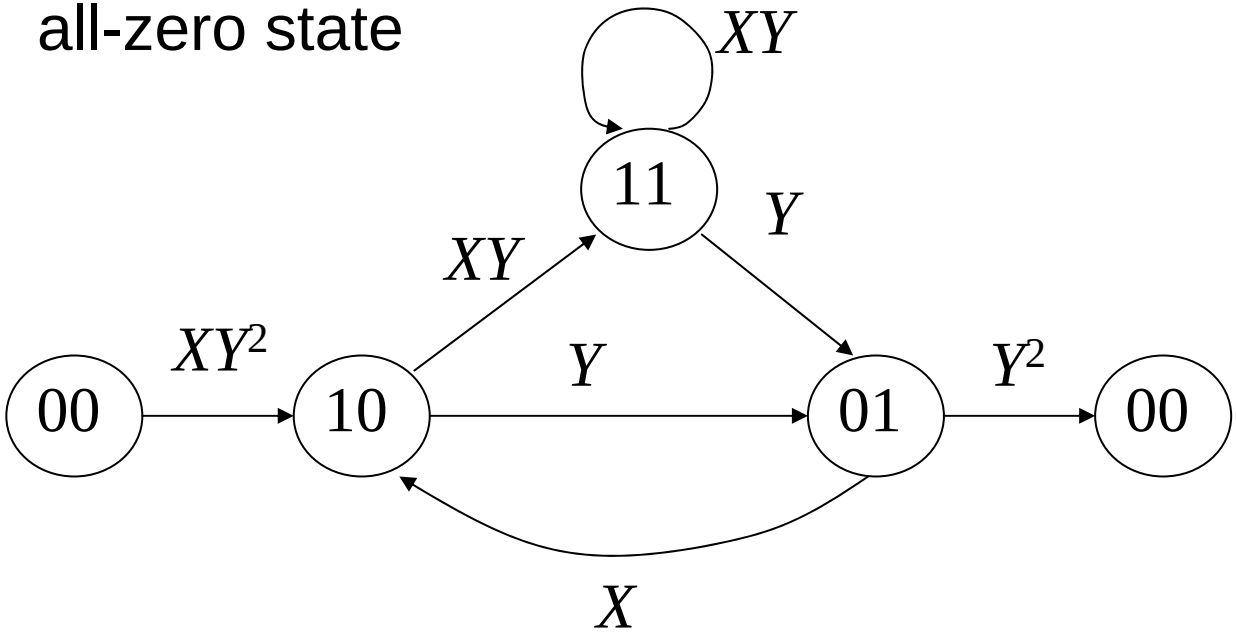
Definition of a non-zero weight path

Any path that diverges and the remerges with the all-zero path



Transfer Function

- Find all the paths that diverge and then remerge with all-zero state



This can be seen as a system with feedforward and feedback loops. The transfer function T(X,Y) is the "gain" of the system.

Mason's Rule

- To find the transfer function of a system with multiple feedforward and feedback loops, we use Mason's rule.

$$T(X,Y) = \frac{\sum_i F_i \Delta_i}{\Delta}$$

- F_i is the gain of the *i*th forward loop. A forward loop goes from start state to end state without passing through a state more than once.
- The graph determinant is Δ. It is given by:

$$\Delta = 1 - \sum_{L_l} C_l + \sum_{L_l L_m} C_l C_m - \sum_{L_l L_m L_n} C_l C_m C_n + \ldots$$

Mason's Rule

- C_l is the gain of the *l*th loop. A loop starts in a state and ends in that same state, without going through any intermediate state more than once.
- L_l and L_m are pairs of non touching loops.
- L_l, L_m, L_n are trios of non-touching loops.
- The cofactor of forward path *i* is Δ_i which is the same as Δ but we eliminate and loops that are touching the *i*th forward loop.

Forward paths

- In our example, we have two forward paths
- Fpath 1 = 00-10-01-00: $F_1 = XY^5$.
- Fpath 2 = 00-10-11-01-00: $F_2 = X^2Y^6$.

Loops

- In our example, the are three loops:
- L_1 = 10 – 01 – 10. $C_1 = XY$
- L_2 = 11-11. $C_2 = XY$
- L_3 = 10-11-01-10. $C_3 = X^2Y^2$

Graph Determinant

- L_1 and L_2 are non touching (they have no states in common). This is the only pair of non-touching loops and there is no set of 3 loops that are non-touching loops.
- The graph determinant is $\Delta = 1-(C_1+C_2+C_3)+(C_1C_2) = 1-2XY-X^2Y^2+X^2Y^2 = 1-2XY$.

Cofactors of Paths 1 and 2

- Fpath 1 does not touch loop 2. Therefore $\Delta_1 = 1\text{-}C_2 = 1\text{-}XY$
- All loops touch Fpath2, therefore $\Delta_2 = 1$.

Transfer Function

$$T(X,Y) = \frac{XY^5(1-XY) + X^2Y^6}{1-2XY} = \frac{XY^5}{1-2XY} =$$

$$XY^5 + 2X^2Y^6 + 4X^3Y^7 + 8X^4Y^8 + \ldots$$

There is one path of weight 5. It is produced by a message of weight 1. There are 2 paths of weight 6 and both are produced by messages of weight 2. There are 4 paths of weight 7 and all are produced by messages of weight 3…