
AUDIO-VIDEO COMPRESSION SYSTEMS: SOME THEORY ELEMENTS

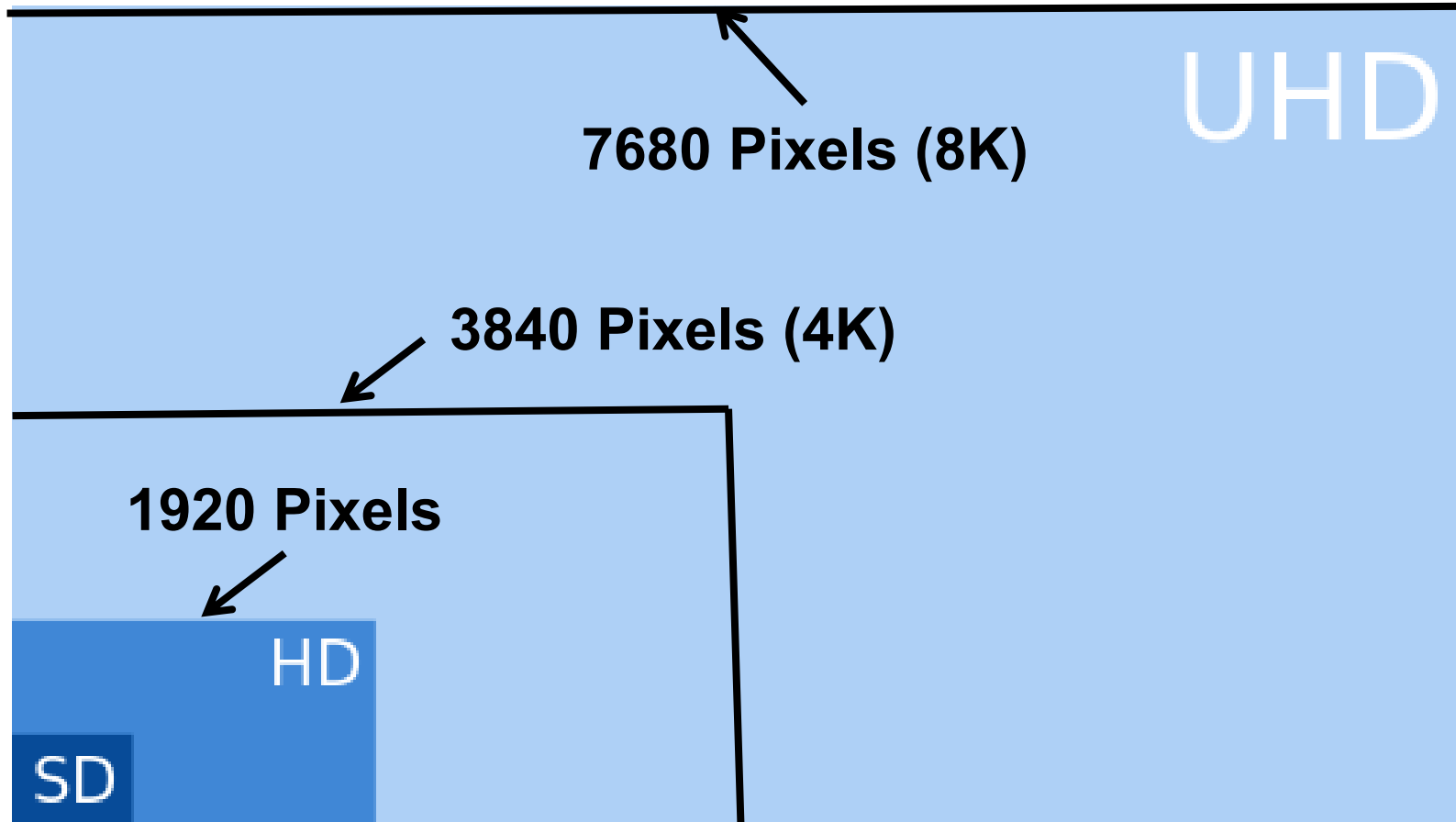
Example: TV – HDTV – UHD TV – 3D Multiview TV

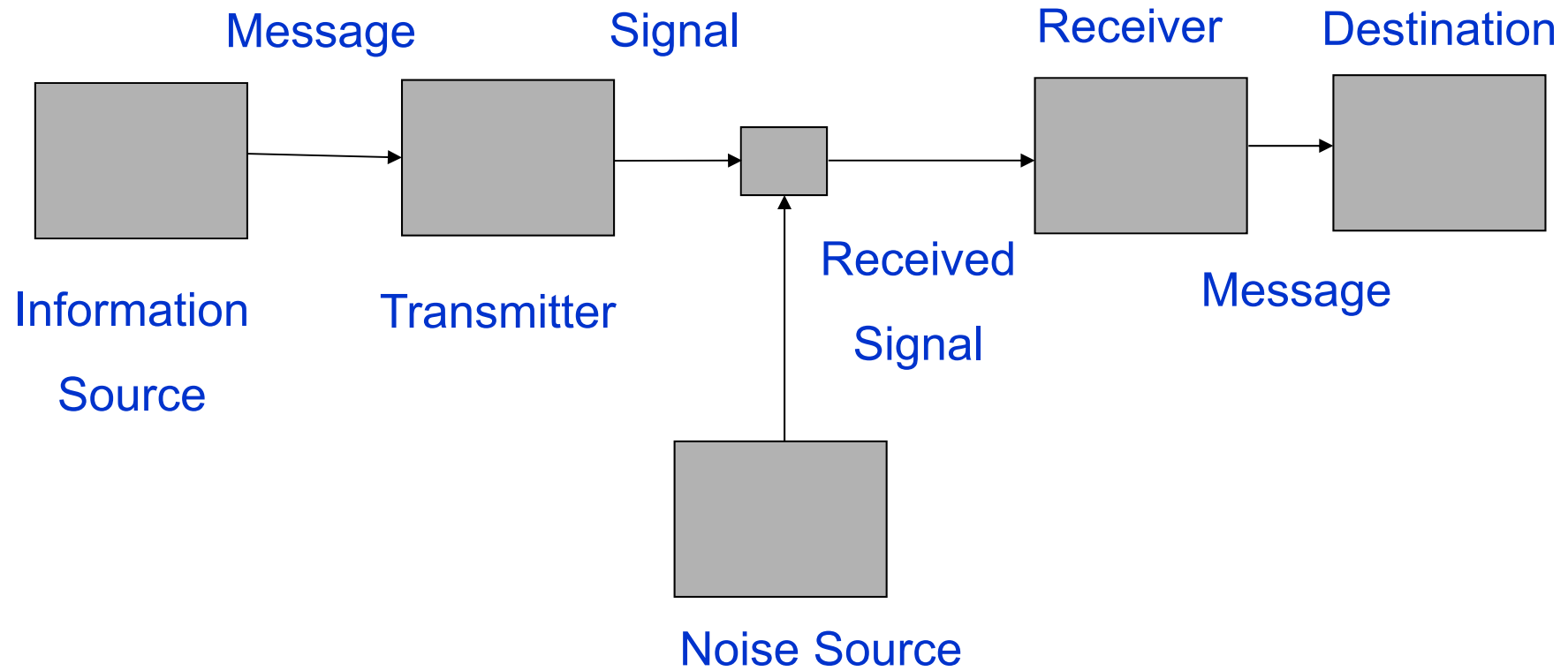
2

- Digital TV (50/60Hz): 169 Mbit/sec (422) 8 b
 - HD-TV (720-P): 0.8 Gbit/sec (422) 8 b
 - HD-TV (1080-P): 1.65 Gbit/sec (422) 8 b
 - 4K UHD TV (3840 × 2160) 50Hz: 6.63 Gbit/sec (422) 8 b
 - 8K UHD TV (7680 × 4320) 50Hz: 26.5 Gbit/sec (422) 8 b

 - Stereo - 3D: Format × 2
 - Multiview - N: Format × N

 - High dynamic range: 10 – 12 – 16 b
 - High framerate: 100 Hz – 120 Hz
 - Extended color space ...
-





Some history: Communication problems (Shannon) ⁵

1. Technical problem:

- What is the accuracy by which the communication signals/symbols can be transmitted?

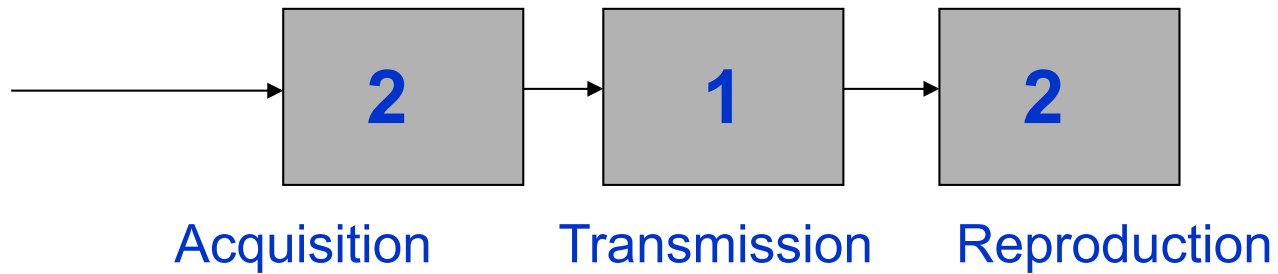
2. Semantic problem:

- What is the accuracy by which the transmitted signals/symbols transport (represent) the desired information

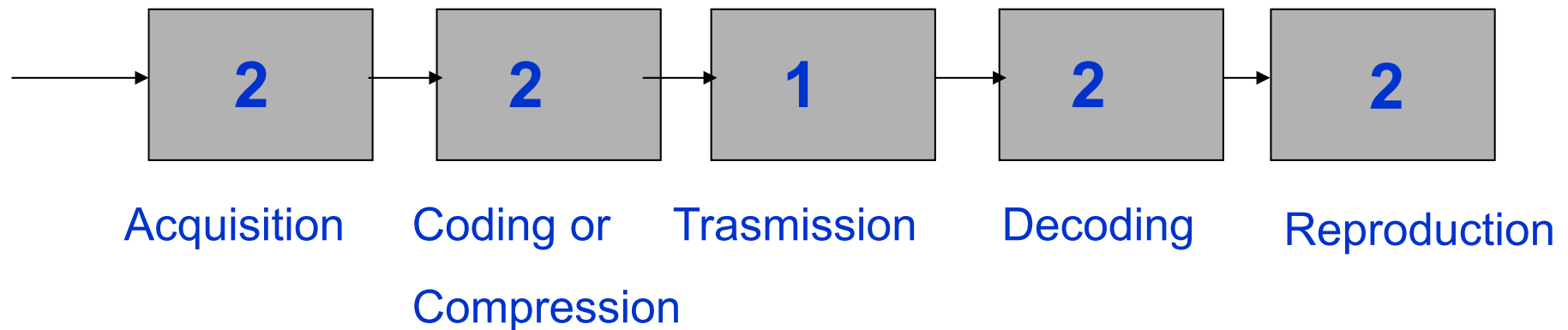
3. Effectiveness problem:

- What is the success of the transmission of the desired meaning?

Analog signals

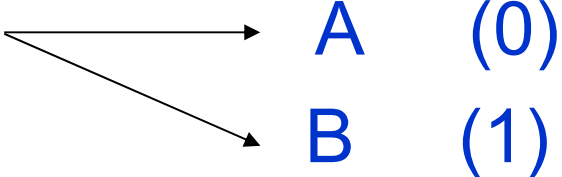


Digital signals



1. How is the quantity of information measured?
2. How the capacity of a transmission channel is measured?
3. The passage from a message to a signal implies a coding. What are the characteristic of an efficient coding?
4. When coding is optimal?
5. If the coding is the most efficient what is the speed at which a channel can transmit the information?
6. What are the effects of noise and its characteristics? How can such effects be minimized?

- There are some theoretical answers for questions 1 and 2 (bounds and theorems).
- For questions 3,4,5,6 the answers are somehow more difficult.
- For question 3 it can be demonstrated that the problem posed is even a non-computable or non-decidable problem.

- The information is a measure of the freedom in the choice of a message:
-  A (0)
B (1)
- The quantity that measures the concept of information results equivalent to the quantity known in thermodynamic theory as « entropy ».

-
- Intuitive properties of the « entropy »:
 - « entropy » can be added
 - « entropy » is proportional to the incertitude of the information source.
 - The quantity of information is proportional to the number of symbols of the source.
 - « entropy » is measured in bit

-
- The quantity of information can be defined as the logarithm of the number of available choices:

1. A 00

2. B 01

3. C 10

4. D 11

$$\log_2 4 = 2$$

information bits

- In reality the information is also related to the probability of each single choice: a less probable choice is capable of transmitting more information.

-
- For these reasons the quantity of information transmitted by a symbol M_i is defined as :

$$I_i = -\log_2 p(M_i)$$

expressed in bit/symbol

-
- Source of information ergodic and without memory that produce different messages M_i associated at one probability p_i :
 - M_1 p_1
 - M_2 p_2
 - M_3 p_3
 -
 - M_n p_n
 - « entropy » (average information generated by an information source: average of the information transmitted by each possible message).

$$H = -[p_1 \log_2 p_1 + p_2 \log_2 p_2 + \dots + p_n \log_2 p_n]$$
$$= -\sum_{i=1}^n p_i \log_2 p_i$$

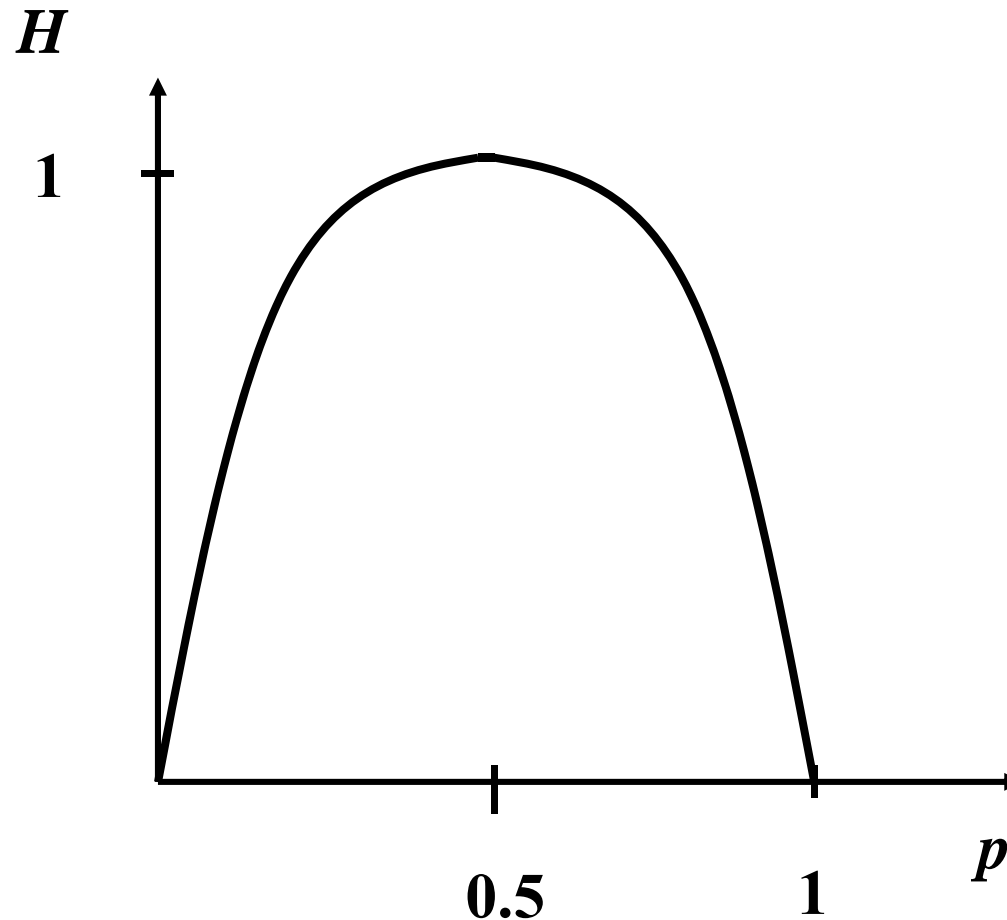
- The entropy of an information source is maximal when all symbols are equiprobable (Kraft inequality).
- $H > 0$, $=0$ only if all the $p_i=0$
- Information source without memory generating a set of n messages M_i identically distributed, the probability of a symbol sequence of length k is:

$$P_{1,2,\dots,k} = \prod_{i=1}^k p_i = \prod_{i=1}^k \frac{1}{n} = \frac{1}{n^k}$$

- Maximal entropy: $p_i = \frac{1}{n}$

Example: 2 messages with prob. p , $(1-p)$ 16

$$H = -p \log_2 p - (1-p) \log_2 (1-p)$$



-
- Fundamental theorem (Shannon):
 - Given a source with entropy H bit/symbols and a channel of capacity C bit/sec without noise, it is possible to code the output of the source in a way to transmit

$$\frac{C}{H} - \varepsilon$$

symbols per second on the channel (with ε as small as possible...).

- Coding problem:
 - Model of the events (definition of the messages) in a way to obtain a source with minimum entropy.
 - Definition of the entropy coder.

-
- Model of the coder consists of two components:
 1. The structure : it specifies the set of events (symbols)
 2. The parameters: the probabilities assigned to each event.The structure of the model aims at capturing the characteristic properties of the “information” generated by the “source”.

The parameters are related to each specific source.
 - Example:
 - Source of information: sampled sinusoidal signal.
 - Structure: sinus wave characterized by amplitude frequency and phase.
 - Parameters: values of amplitude frequency and phase.
-

Suppose we wish to code the string of 10^9 digits starting with:
"314159265358979323846264..."

Assume our model is that each digit occurs with probability 0.1, independent of any other digits.

The entropy of each symbol (digit) is about 3.3219....

We would need to transmit about 3.321×10^9 bits (Shannon)

Assume another "Model": the string represents the first 10^9 digits of π :

Transmit the pseudo code program to calculate π and the upper limit (10^9) = 52 + 8 bytes!!

- Given a source of symbols non ergodic, non stationary, and with memory, what is the entropy of such source?
- Infinite order statistics (not even possible for a finite set of data).

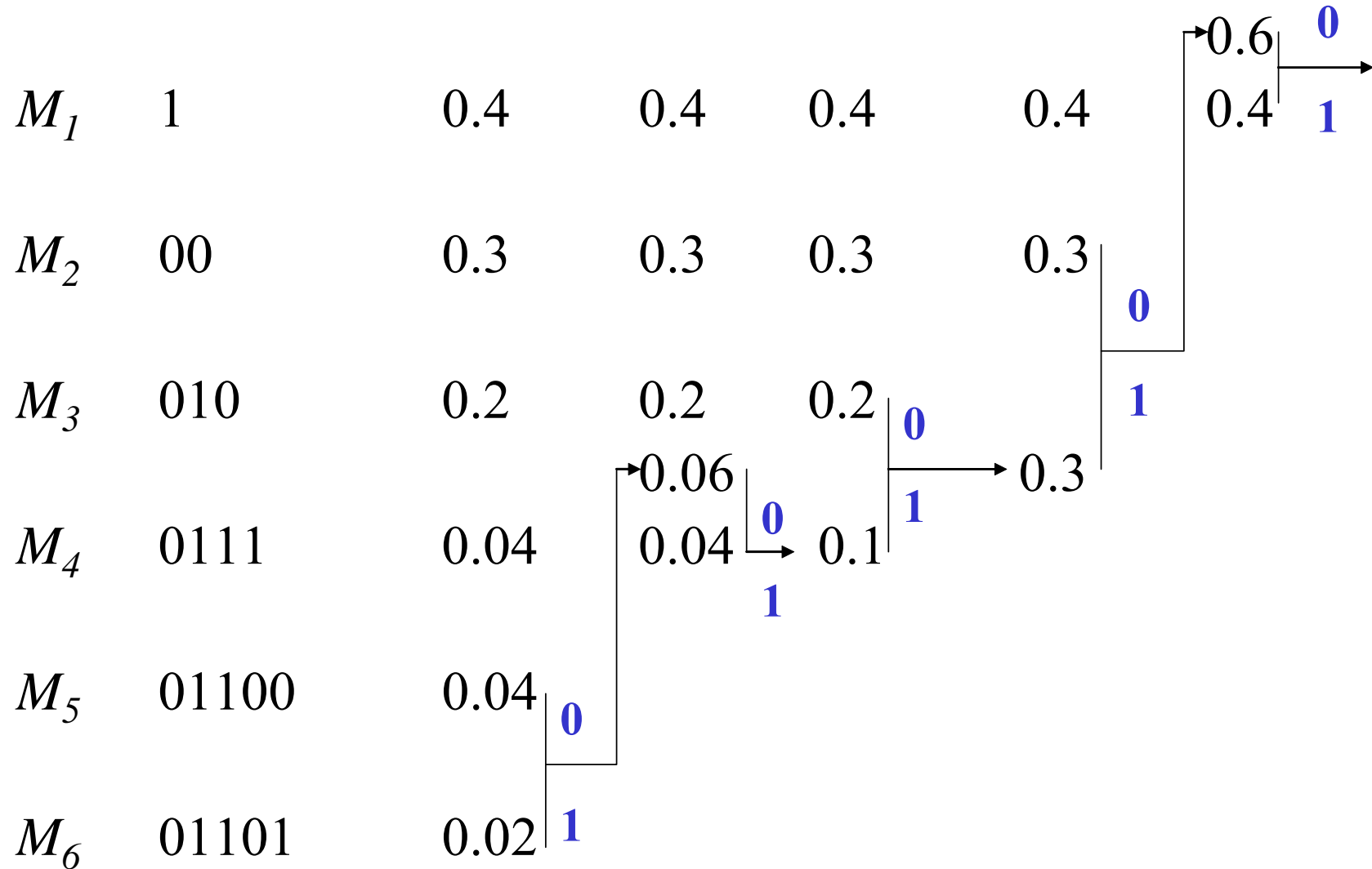
$$H_{\infty} < H^m < H_0$$

- The problem is to find the “model” m for which the entropy H^m is the lowest.

-
- It can be shown that such problem cannot be solved as minimization of a « functional ». The class of the « models » is not computable: the problem is « non-decidable ».
 - Solution: propose a model and verify if the corresponding entropy is lower than other known models (efficiency of the model).

- The function of an entropy coder: to transmit for each symbol (message) a quantity of bits proportional to the information carried by the transmitted symbol (i.e. to transmit a number of bits equal to the entropy of the source).
- Huffman coding
- Arithmetic coding

code prob₁ prob₂ prob₃ prob₄ prob₅



- Huffman coding efficiency:

- Example, source of two messages:

a (sun), b (rain)

$$P(a) = p_1 = 0.8, \quad P(b) = p_2 = 0.2$$

$$\text{Entropy } H = -p_1 \log_2 p_1 - p_2 \log_2 p_2 = 0.72 \text{ bit/symbol}$$

Huffman coding:

a 1 b 0 1 bit/symbol

- Alphabet extension

aa 0 0.64

ab 11 0.16 **0.78 bit/symbol**

ba 100 0.16

bb 101 0.04

- Example of previous slide (sun, rain)

M_1	1	0.4
M_2	00	0.3
M_3	010	0.2
M_4	0111	0.04
M_5	01100	0.04
M_6	01101	0.02

Entropy $H=1.999$

Huffman coding = **2.06 bit/symbol**

- The inefficiency of the Huffman coder:
 - It always assigns an integer number of bits for each symbol = $-\log_2 p_i$
 - In the past it was always solved by means of alphabet extensions trying to approximate the optimal probability distributions in (Huffman sense)
 - Optimal distribution:
with n integer.
- Theorem (Rissanen 1984): models that use whatever type of alphabet extensions are inferior to the best model that does not use any alphabet extension.

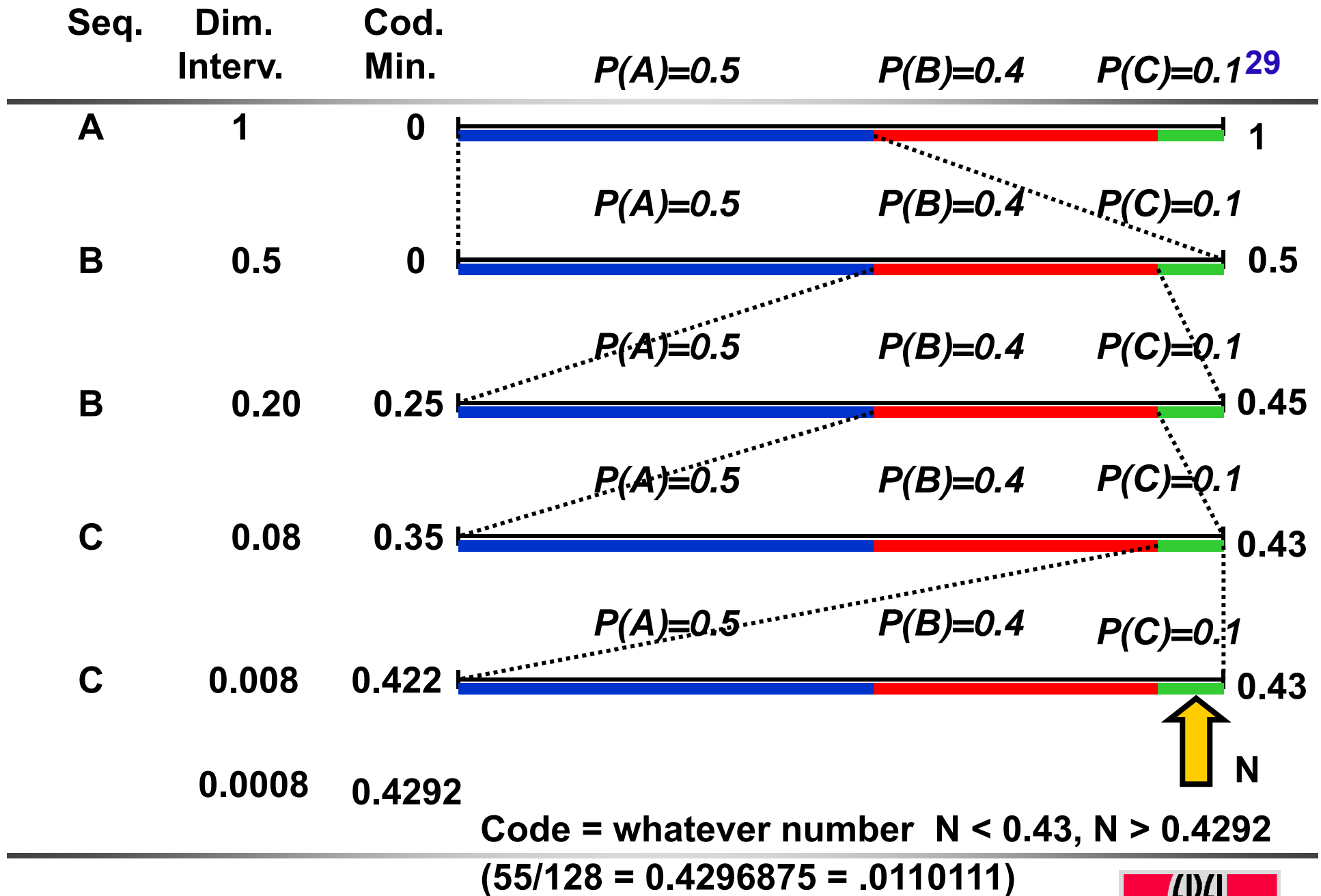
$$p_i = \frac{1}{2^n}$$

-
- Problem: does it exist an entropic coder that assign for each model and parameter a fractional number of bits/symbols?
 - Arithmetic coder (Rissanen, Langdon 1984)
 - The basic idea of the arithmetic coder is to represent a sequence of symbols with a real number.
 - In practice the idea is to represent a sequence of symbols by an association with a binary representation of an interval of length less than 1.


Arithmetic Coding

$P(A) = 1/3$ $P(B) = 2/3$

		AAA ↓		Segment	Code
1	A 8/9	AA	AAB	31/32	.11111
			ABA	15/16	.1111
		AB	ABB	14/16	.1110
				6/8	.110
2/3	B 4/9	BA 16/27	BAA	10/16	.101
			BAB	4/8	.100
		BB 8/27	BBA	3/8	.011
			BBB	1/4	.01



Decoding: code N = 55/128 Interval Dim. = 1, Cod. Min. = 0

$55/128 < 1 P(A)$  first symbol = A

Interval Dim. = 0.5, Cod. Min = 0

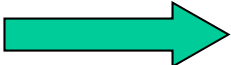
$55/128 > 0.5 P(A)$ and $< 0.5 (P(A)+P(B))$  second symbol = B

Interval Dim. = $0.5 (P(A)+P(B)) - 0.5 P(A) = 0.20$ Cod. Min. = 0.25

$55/128 > 0.25 + 0.20 P(A)$ and $< 0.25 + 0.20(P(A)+P(B))$

 third symbol = B

Interval Dim. = $0.20 (P(A)+P(B)) - 0.20 P(A) = 0.08$ Cod. Min. = 0.35

$55/128 > 0.35 + 0.08 (P(A)+P(B))$  fourth symbol = C

.....

ARITHMETIC CODING

- Recursive processing: (**Cod. Min**), (**Interval Dim.**)

For each received symbol (a_i):

New **Cod. Min** = **Cod. Min** + (**Interval Dim.**) P_i

New **Interval Dim.** = (**Interval Dim.**) p_i

P_i = cumulative probability a_i (**Model and Probability are known**)

p_i = probability a_i

CODER:

STEP 0: **Cod. Min. = 0, Interval Dim. = 1**

STEP 1: receive symbol from source, (if end of symbols,
symbol = **EOF**), calculate new **Cod. Min e Interval Dim.**

STEP 2: **IF** symbol **EOF**, transmit the string of bits that
identifies a number included into the current interval, **STOP**

ELSE transmit the string which is univocally defined at the
current moment, **GOTO STEP1**

DECODER

Model and probabilities of the coder are known

STEP 0: Cod. Min = 0, Interval Dim. Interval = 1

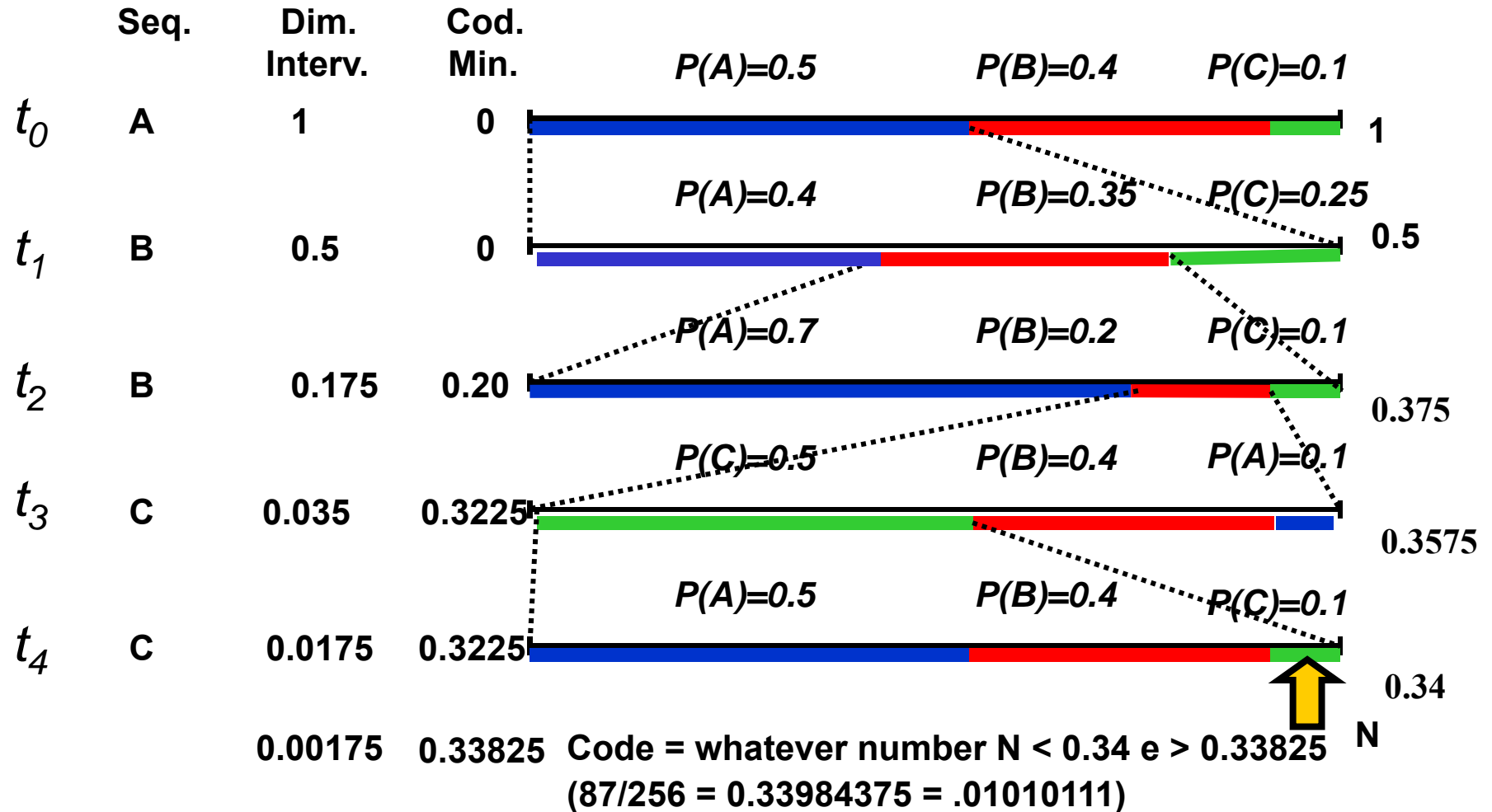
STEP 1: compare the received string and
determine the corresponding interval,

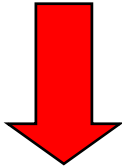
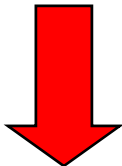
STEP 2: IF one symbol can be decoded decode it
ELSE GOTO STEP 1

STEP 3: IF symbol is **EOF STOP**
ELSE calculate new **Cod. Min.** and
Interval Dim. GOTO STEP 2

Non-stationary or adaptive probability model:

$$P_A(t_i), \quad P_B(t_i), \quad P_C(t_i) \quad i = 0, \dots, K \quad 33$$



-
- The choice of the entropy coder can be done « A Priori » if only the performance and the coding characteristics are considered:
 - VLC (Huffman coding):
 - Optimal efficiency only for some probability distributions ($1/2^n$) (never less than one bit per symbol, always an integer number of bits per symbols)

 - May result complex for adaptive and non-stationary models (large memories for several tables of VLC symbols)


– Arithmetic coding

35



+ Optimal for whatever probability distribution



+ easy to be implemented also for adaptive and complex source models.

- The choice of the type of entropy coder depends on the system specifications:
 - Overall complexity (model + entropy coder)
 - Granularity for the data access, multiresolution for the rapid scanning of data
 - Robustness versus transmission errors, re-synchronization properties.
 - Bitrate control

VLC (Huffman coding):

36

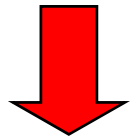


+ Simple coder (for non-adaptive source models)

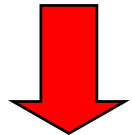


+ Self synchronizing code (it is relatively robust to transmission errors, the random access to data is simpler)

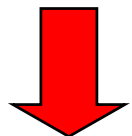
– Arithmetic coding



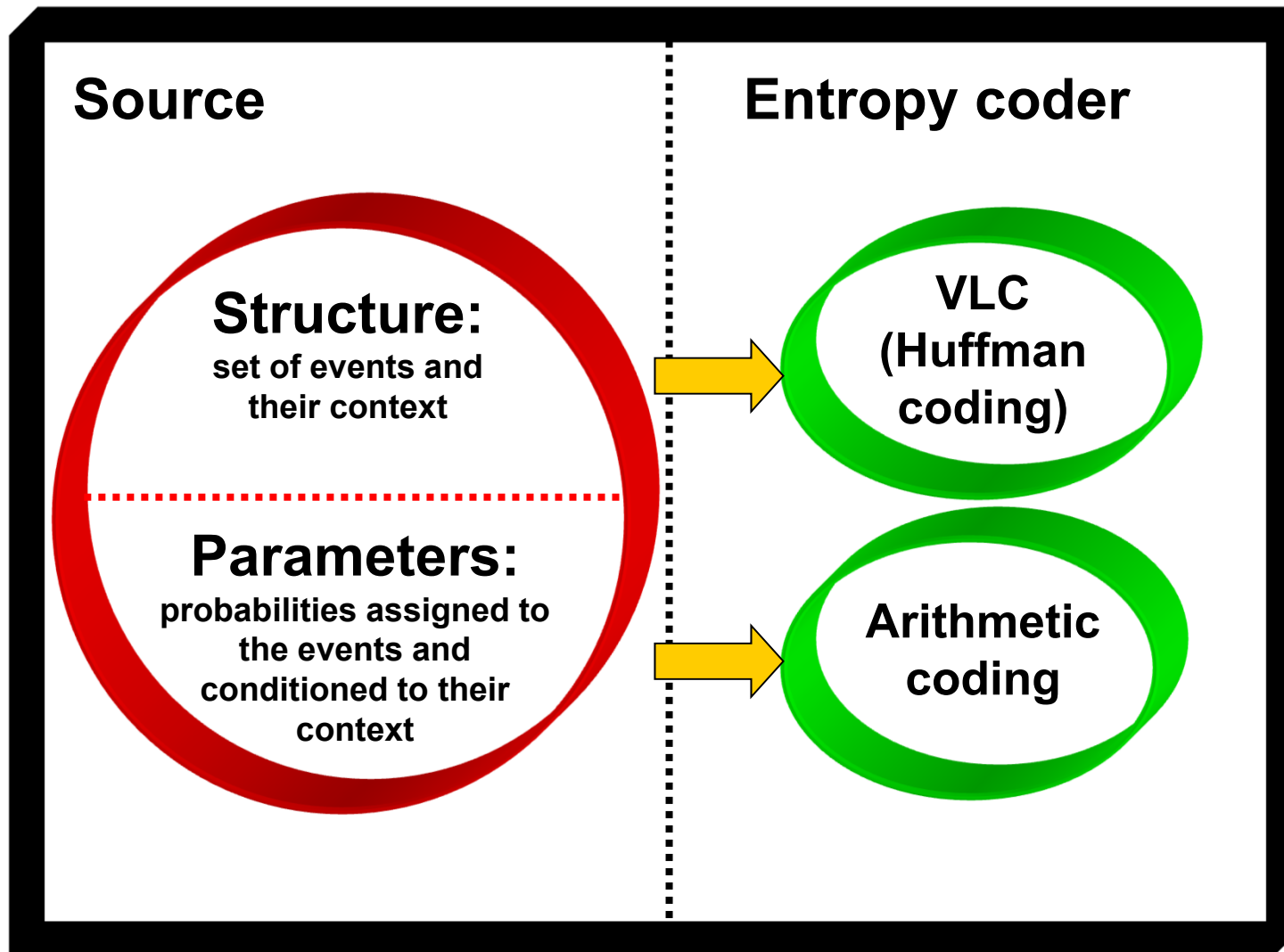
- Need to process large blocks of data



- It is not robust to transmission errors



- It is more complex to implement in the case of non-adaptive source models (multipliers are necessary)



- Information theory places hard limits on what can and cannot be compressed losslessly, and by how much:
 - 1. There is no such thing as a "universal" compression algorithm that is guaranteed to compress any input, or even any input above a certain size. In particular, it is not possible to compress random data or compress recursively.
 - 2. Given a model and its parameters (probability distribution) of your input data, the best you can do is code m symbols with probability p using $\log_2 1/p$ bits. Efficient and optimal codes are known (Huffman/Arithmetic).

-
- Compression = modeling + coding.
 - Coding is a solved problem. Modeling is provably not solvable.
 - The entropy coder are always lossless!!!
 - The difference between lossy and lossless coding is only in the definition of the structure of the model (definition of the set of events or symbols)

- Example of adaptive coder that builds during the coding itself the model of the information source:
 - Coding of variable length symbols into fixed length coding symbols
 - The tables are built in « real time »
 - $\{a,b,c,\dots\}$ n symbols
 - Table of dimension k con $k \gg n$ ($k=2^j$)

• Symbols	Codes
a	1
b	2
c	3
...	...
...	n
ab	n+1
ba	n+2
...
aa
...
aaaaa	k

Example

Compress the string
"TOBEORNOT
TOBEORTOB
EORNOT#"

Symbol	Binary	Decimal	Symbol	Binary	Decimal
#	00000	0	N	01110	14
A	00001	1	O	01111	15
B	00010	2	P	10000	16
C	00011	3	Q	10001	17
D	00100	4	R	10010	18
E	00101	5	S	10011	19
F	00110	6	T	10100	20
G	00111	7	U	10101	21
H	01000	8	V	10110	22
I	01001	9	W	10111	23
J	01010	10	X	11000	24
K	01011	11	Y	11001	25
L	01100	12	Z	11010	26

Ziv-Lempel Coder

43

Current Sequence	Next Char	Output Code	Output Bits	Extended Dictionary	Comments
Null	T				
T	O	20	10100	27: TO	27 = first available code after 0 through 26
O	B	15	01111	28: OB	
B	E	2	00010	29: BE	
E	O	5	00101	30: EO	
O	R	15	01111	31: OR	
R	N	18	10010	32: RN	32 requires 6 bits, so for next output use 6 bits
N	O	14	001110	33: NO	
O	T	15	001111	34: OT	

Ziv-Lempel Coder

44

Current Sequence	Next Char	Output Code	Output Bits	Extended Dictionary	Comments
T	T	20	010100	35: TT	
TO	B	27	011011	36: TOB	
BE	O	29	011101	37: BEO	
OR	T	31	011111	38: ORT	
TOB	E	36	100100	39: TOBE	
EO	R	30	011110	40: EOR	
RN	O	32	100000	41: RNO	
OT	#	34	100010		# stops the algorithm; send the cur seq
		0	000000		and the stop code

- Original non encoded (compressed) length = 25 symbols
× 5 bits/symbol = 125 bits
- Encoded (compressed) length = (6 codes × 5 bits/code) +
(11 codes × 6 bits/code) = 96 bits.
- Using LZW has saved 29 bits out of 125, reducing the
message by almost 22%.
- If the message were longer, then the dictionary words
would begin to represent longer and longer sections of
text, allowing repeated words to be sent much more
compactly.

Ziv-Lempel Decoder

46

Input Bits	Input Code	Output Sequence	New Dictionary Entry: Full	New Dictionary Entry: Conjecture	Comments
10100	20	T		27: T?	
01111	15	O	27: TO	28: O?	
00010	2	B	28: OB	29: B?	
00101	5	E	29: BE	30: E?	
01111	15	O	30: EO	31: O?	
10010	18	R	31: OR	32: R?	created code 31 (last to fit in 5 bits)
001110	14	N	32: RN	33: N?	so start reading input at 6 bits
001111	15	O	33: NO	34: O?	
010100	20	T	34: OT	35: T?	

Ziv-Lempel Decoder

47

Input Bits	Input Code	Output Sequence	New Dictionary Entry: Full	New Dictionary Entry: Conjecture	Comments
011011	27	TO	35: TT	36: TO?	
011101	29	BE	36: TOB	37: BE?	36 = TO + 1st symbol (B) of
011111	31	OR	37: BEO	38: OR?	next coded sequence received (BE)
100100	36	TOB	38: ORT	39: TOB?	
011110	30	EO	39: TOBE	40: EO?	
100000	32	RN	40: EOR	41: RN?	
100010	34	OT	41: RNO	42: OT?	
000000	0	#			

- At each stage, the decoder receives a code X ; it looks X up in the table and outputs the sequence χ it codes, and it conjectures $\chi + ?$ as the entry the encoder just added – because the encoder emitted X for χ precisely because $\chi + ?$ was not in the table, and the encoder goes ahead and adds it.
- But what is the missing letter? It is the first letter in the sequence coded by the *next* code Z that the decoder receives.
- So the decoder looks up Z , decodes it into the sequence ω and takes the first letter z and tacks it onto the end of χ as the next dictionary entry.
- This works as long as the codes received are in the decoder's dictionary, so that they can be decoded into sequences.

- What happens if the decoder receives a code Z that is not yet in its dictionary? Since the decoder is always just one code behind the encoder, Z can be in the encoder's dictionary only if the encoder *just* generated it, when emitting the previous code X for χ . Thus Z codes some ω that is $\chi + ?$, and the decoder can determine the unknown character as follows:
 1. The decoder sees X and then Z .
 2. It knows X codes the sequence χ and Z codes some unknown sequence ω .
 3. It knows the encoder just added Z to code $\chi +$ some unknown character,
 4. and it knows that the unknown character is the first letter z of ω .
 5. But the first letter of ω ($= \chi + ?$) must then also be the first letter of χ .
 6. So ω must be $\chi + x$, where x is the first letter of χ .
 7. So the decoder figures out what Z codes even though it's not in the table,
 8. and upon receiving Z , the decoder decodes it as $\chi + x$, and adds $\chi + x$ to the table as the value of Z .

-
- **STEP 0:** the first n values of the table are the symbol alphabet, the current string S is empty,
 - **STEP 1:** **IF** received a symbol P append P to S , consider SP as current string, **ELSE** transmit a word of j bit corresponding to the current string, **STOP**.
 - **STEP 2:** **IF** SP is present in the table, assign SP to the current string, **ELSE** (SP is not in the table)
 - **Step 2.1:** transmit a word of j bit for S
 - **Step 2.2:** if there is an empty row in the table add SP to the table
 - **Step 2.3:** assign P to the current string, **GOTO step1**.

-
- **Pros:**
 - It approaches the entropy of the source (for a memory of infinite size)
 - It does not need to define an explicit model structure and to estimate the relative parameters (probabilities)
 - Implicitly considers the statistical dependencies (conditioned probabilities)
 - It is efficient for non stationary sources (variable probabilities from source to source, i.e. Cod. ASCII of different languages)
 - Simple coding and decoding
 - Fixed length coding (synchronization and error correction)
-

-
- Disadvantages:
 - Complexity (dimension of the table and search process)
 - Slow to converge for local statistical variations.
 - Variants:
 - Reconstruction of the table every XXX input symbols
 - Replace the table elements that have not been used for the last XXX input symbols
 -

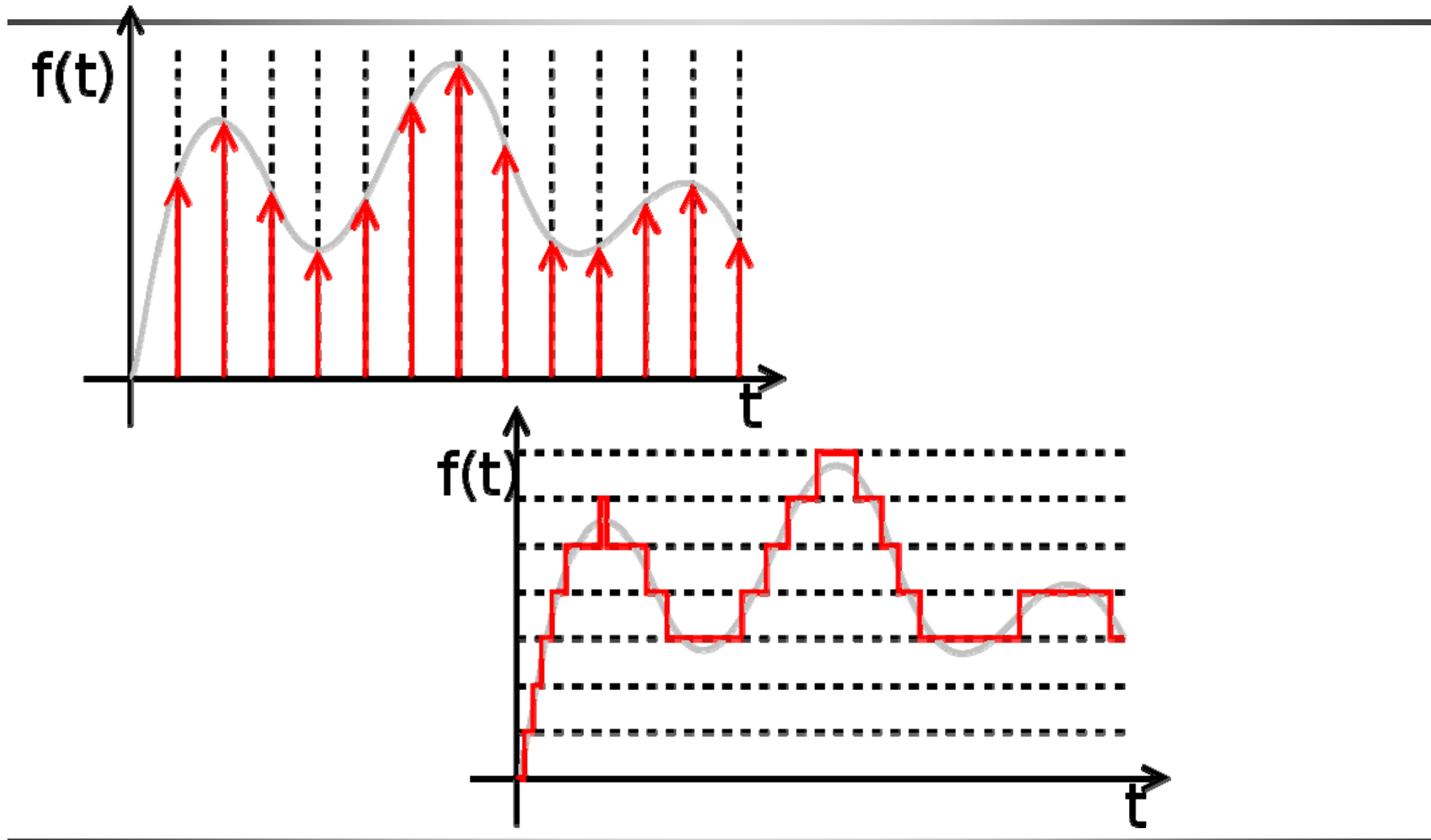
- Inefficiency of the coding caused by the probability estimation error (« information divergence » or « Kullback distance »):
 - p_i^e (estimated probabilities)
 - p_i^r (real probabilities)

$$D(p^e, p^r) = \sum_i p_i^r \log_2 \frac{p_i^r}{p_i^e}$$

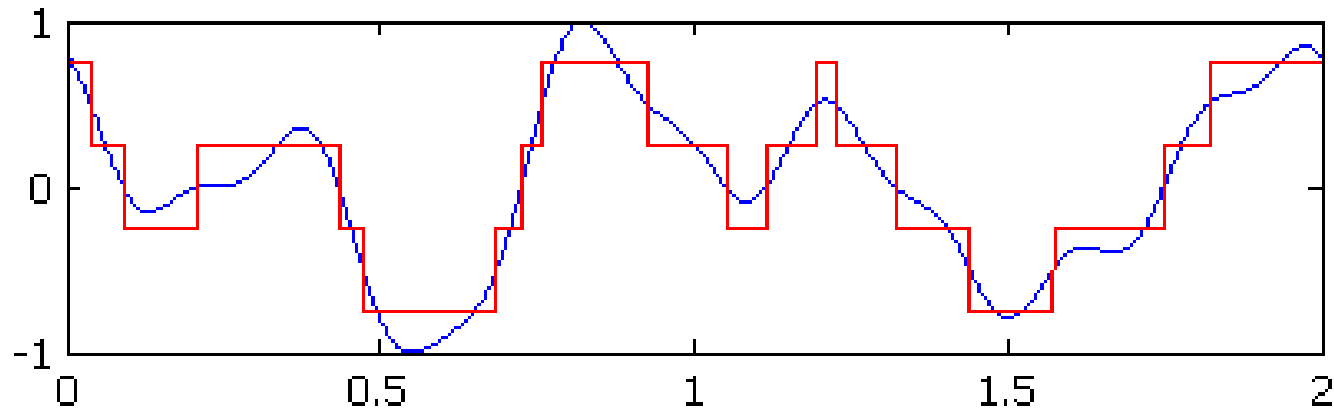
$$H^e = H^r + D(p^e, p^r)$$

- $D(\dots) \geq 0$
- It can be show that $D(\dots)=0$ only if:
 - $p^e_i = p^r_i$

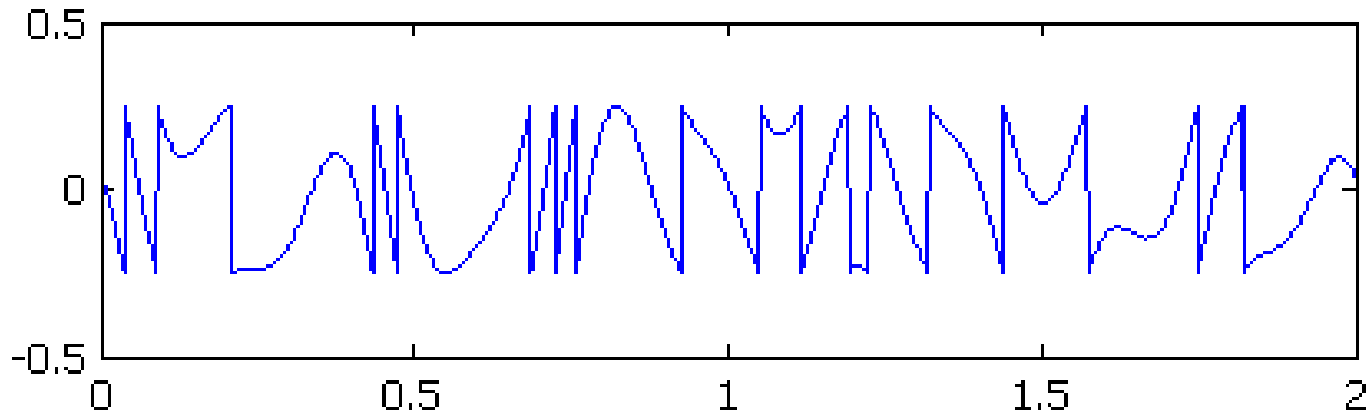
- Do implementations of arithmetic coding exist that do not require multipliers?
- Under which assumptions/limitations can be defined?
 - On the source model?
 - On the implementation itself?
 - For which alphabet sizes?
- The answer next lesson

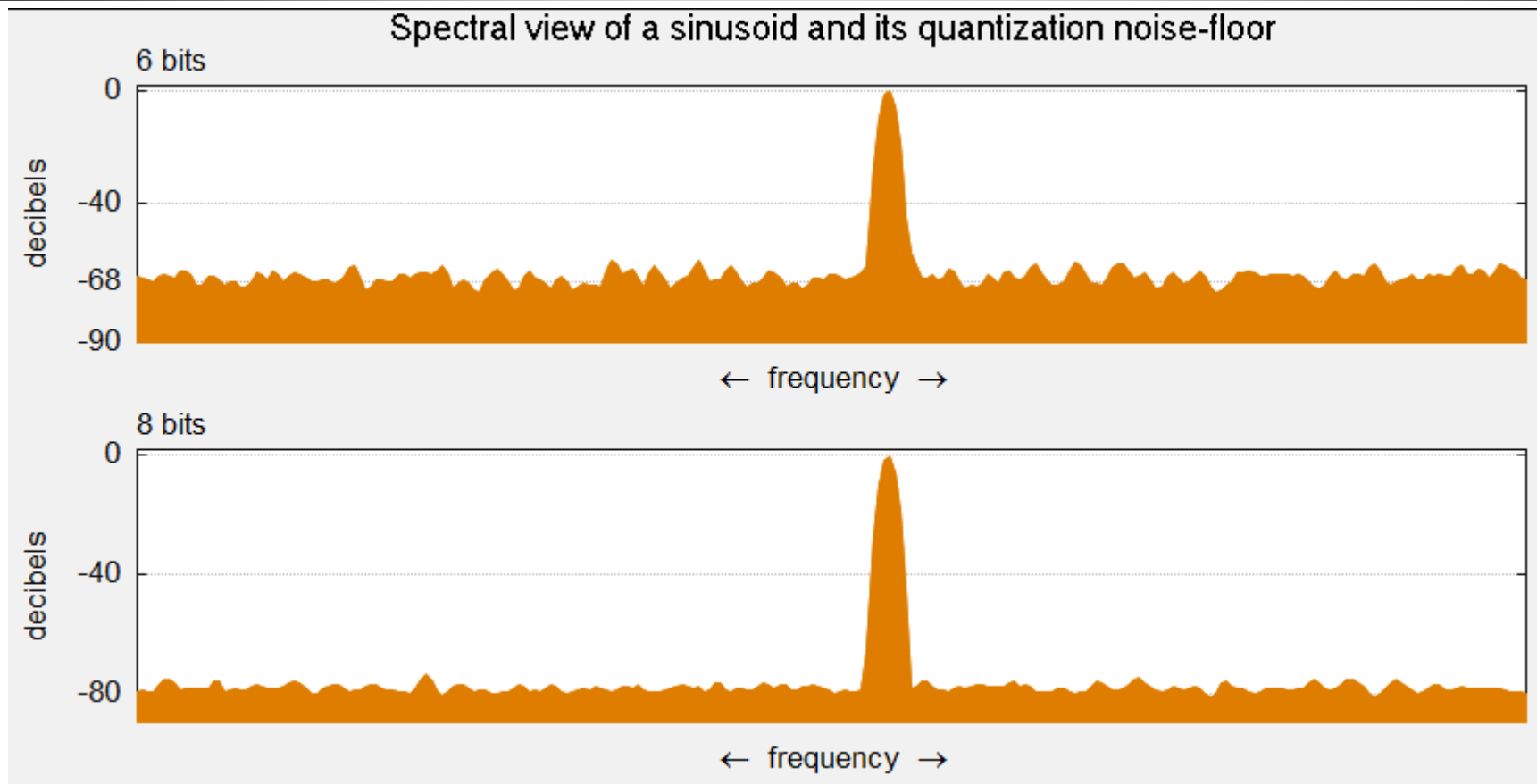


Original and Quantized Signal



Quantization Error





Standard deviation = $(1/\sqrt{12})\text{LSB} \sim 0.289 \text{ LSB}$

Sign-to-Quant-Noise-Pow-Ratio = $10\text{Log}(4) = 6.02 \text{ dB per bit}$

- **Lossy models: definition of set of events (symbols), quantization noise.**
- **Problem: optimal quantization? (that minimize the loss introduced in the information source by the model definition assuming an appropriate criterion, quadratic error for instance).**
- **In the field of coding the problem is different: the optimal quantization should minimize the entropy of the samples (« symbols ») generated for a given distortion of the information.**
- **Non uniform quantization:**
 - **High probability: high density of quantization levels**
 - **Low probability: low density of quantization levels.**
- **Result: probability distribution of symbols roughly uniform.**
- **Uniform distribution: maximum entropy!**
- **Optimal quantization (that minimize H): « Max-Lloyd » in practice is very similar to a uniform quantizer.**

- Transform coding (linear) :
 - Monodimensional (audio) e bidimensional (images):
 - Discrete Fourier Transform
 - Discrete Cosine transform (DCT)
 - Walsh-Hadamard Transform
 - Karhunen-Leuwe Transform (KLT)
 - Wavelet Transform

- Reversible transforms between vectors :
 - To concentrate the energy in fewer coefficients
 - Statistical distribution of coefficients highly non uniform: better suited for entropy coding.
 - High probability of several null value coefficients (EOF symbol)
 - Quantization of Transform coefficients can be made by means of psycho-visual or psycho-acoustic criteria

- Real part of the FFT:
 - Applied on data blocks (8x8) (16x16) (32x32) (64x64).
 - Used by all video coding standards (H.261, H.263, MPEG-1, MPEG-2, MPEG-4 AVC, HEVC)

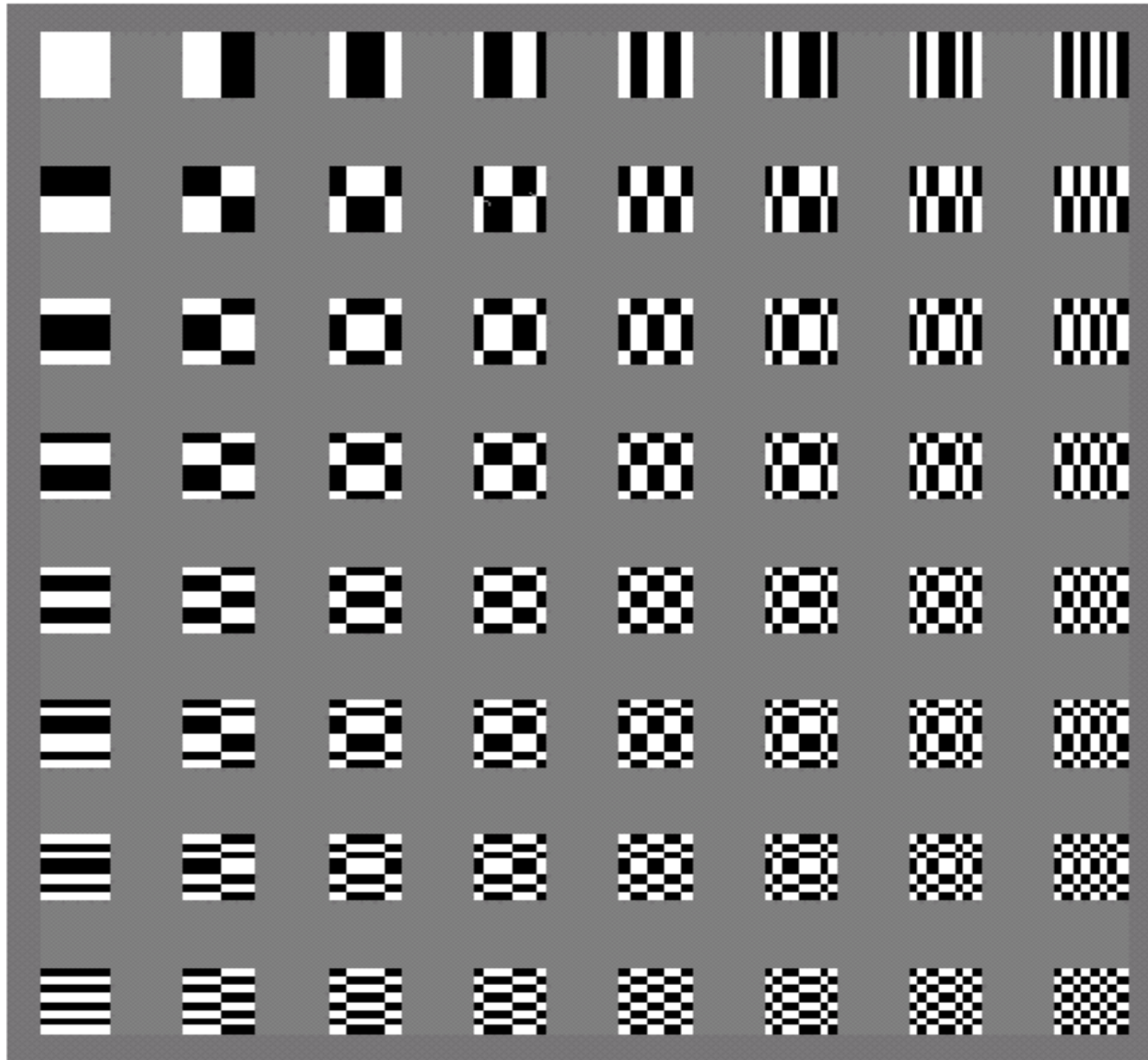
$$F(u, v) = c(u)c(v) \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} f(i, j) \cos \frac{(2i+1)u\pi}{2N} \cos \frac{(2j+1)v\pi}{2N}$$

$$f(i, j) = c(u)c(v) \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} F(u, v) \cos \frac{(2i+1)u\pi}{2N} \cos \frac{(2j+1)v\pi}{2N}$$

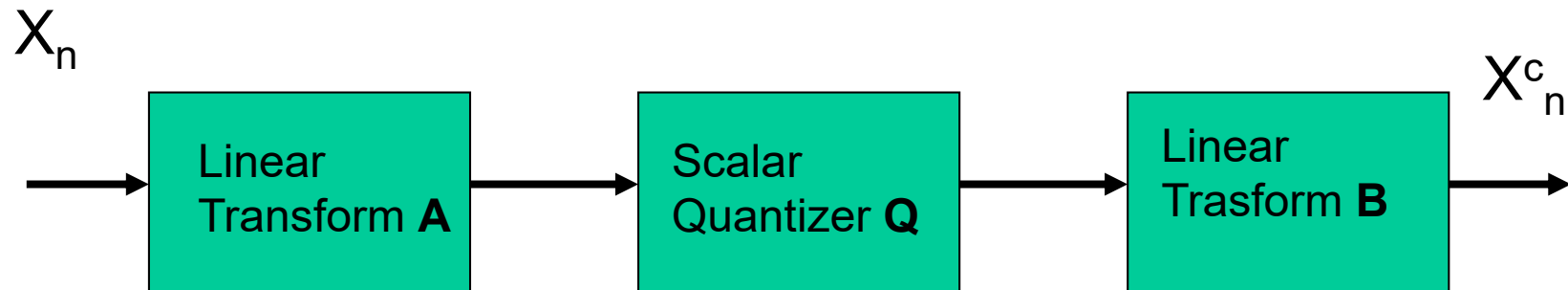
$$i, j, u, v = 0, 1, \dots, N-1$$

$$c(\alpha) = \frac{1}{\sqrt{N}}, \alpha = 0 \quad c(\alpha) = \frac{\sqrt{2}}{\sqrt{N}}, \alpha \neq 0$$

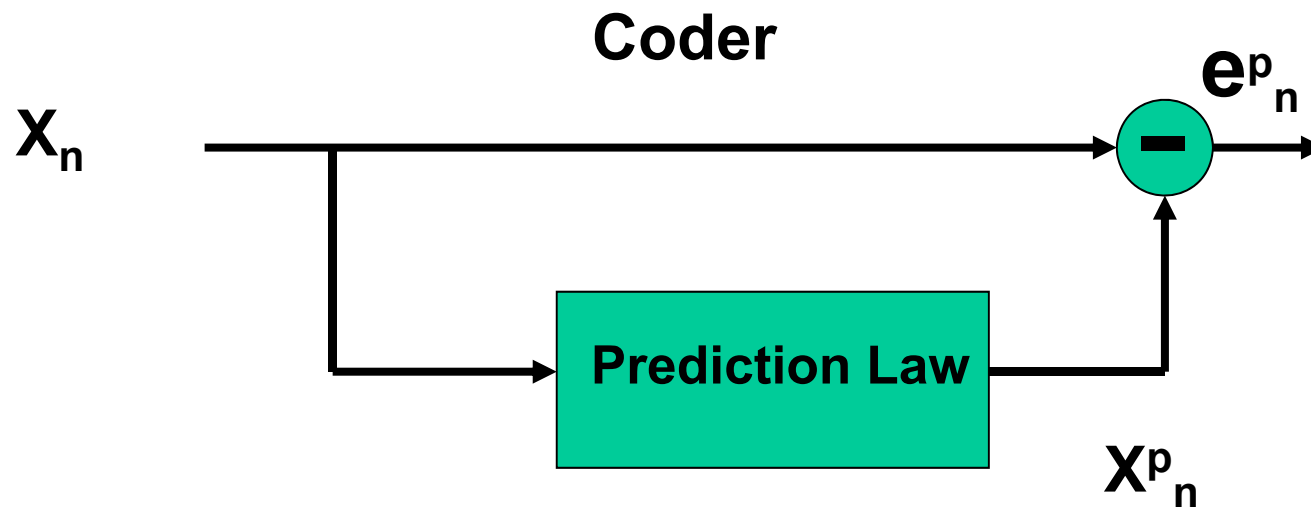
- Separable (8/16/32/64 x1) horizontally and vertically
- Fast software and hardware implementations!

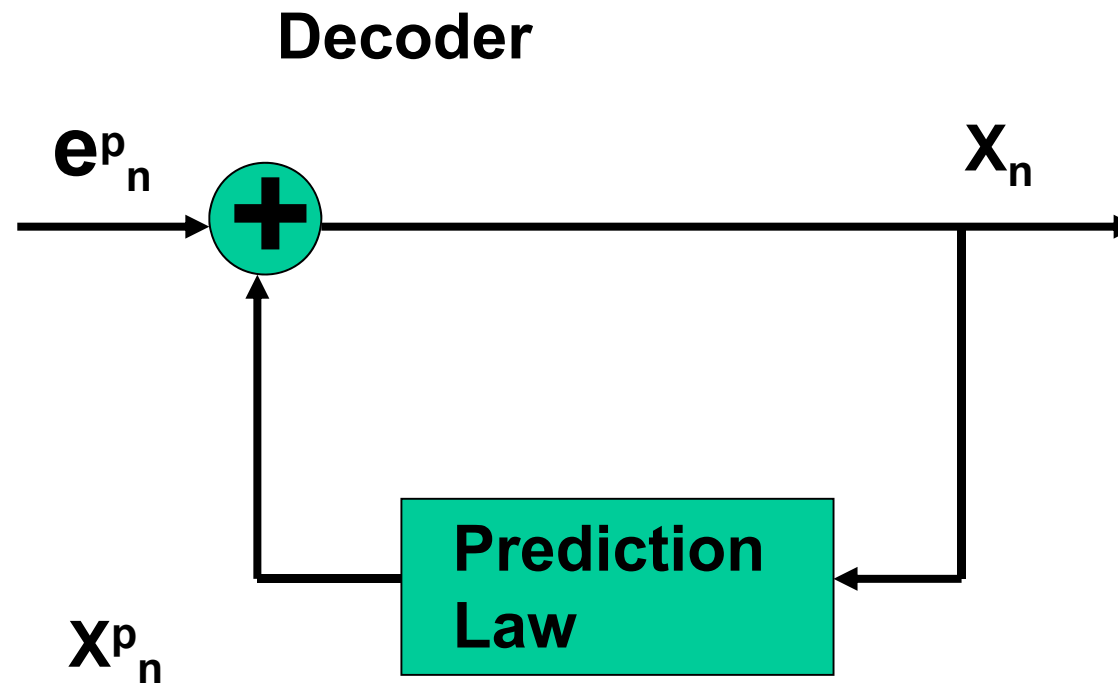


-
- KLT transform:
 - Optimal transform all coefficients are decorrelated
 - The transform is input data dependent
 - If x Gaussian the coefficient are independent
 - Not useful for coding
 - DCT
 - Near to optimality (DCT approximate the KLT)
 - It is independent from input data
 - It is simple to implement
 - Walsh-Hadamard
 - Seldom used for coding (recently for 2x2 and 4x4 blocks of MPEG-4 AVC/H.264)



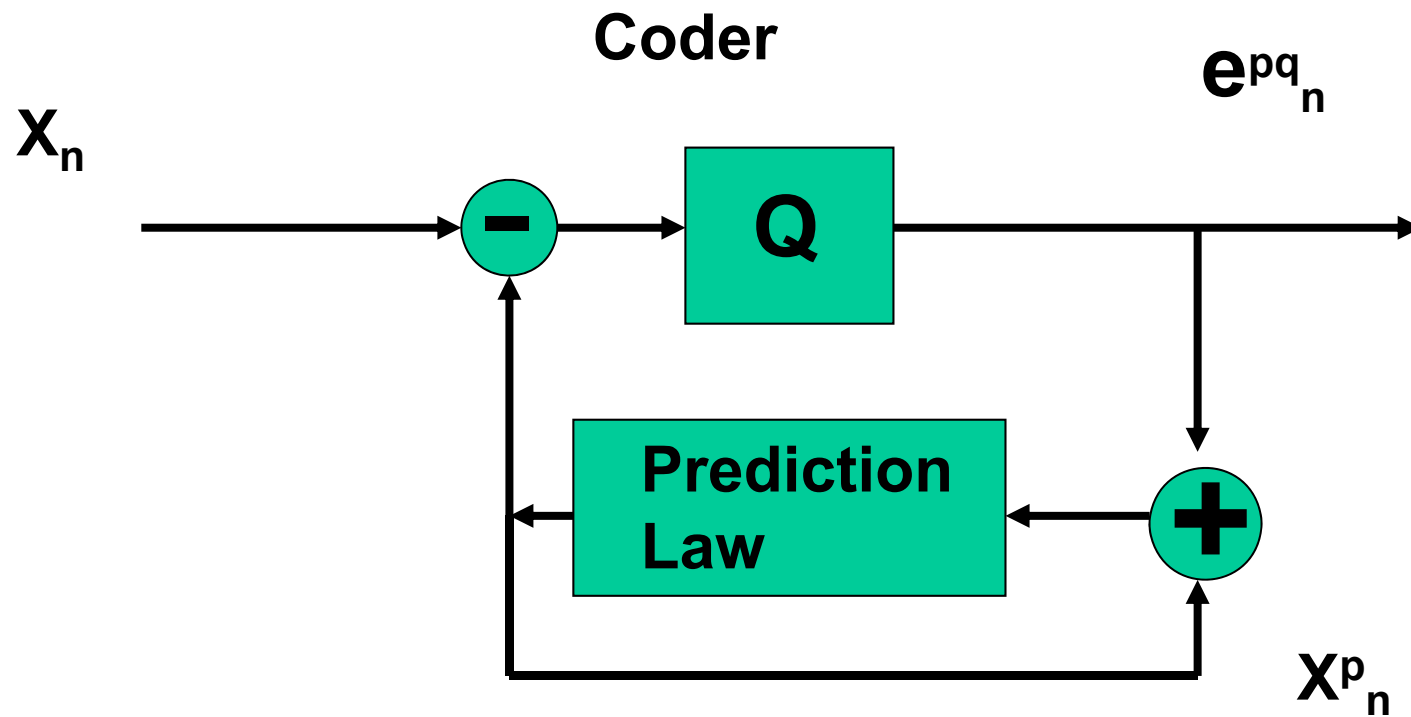
- Objective:
 - Minimize the mean square error $(X_n - X_n^c)$
- Optimal: **A, Q, B?**
- Solution:
 - $A = \text{KLT of } X_n$
 - $Q = \text{Max-Lloyd quantizer}$
 - $B = A^{-1}$





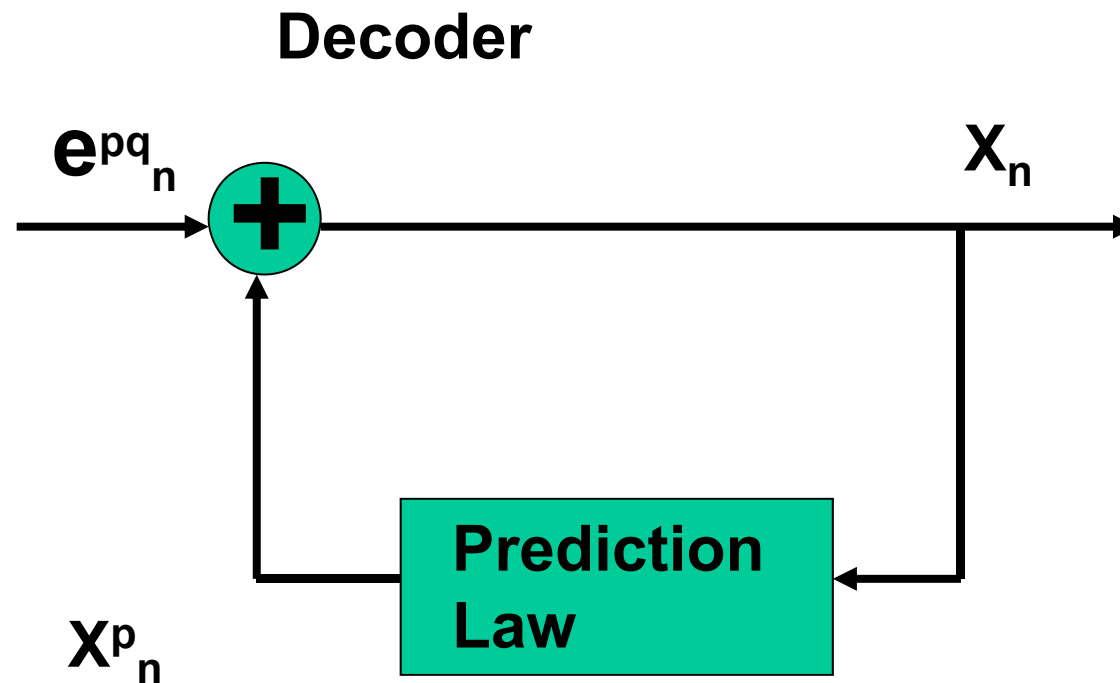
Prediction coding with quantization (non-reversible coding encoder)

69



Prediction coding with quantization (non-reversible coding Decoder)

70



- H.261 (Videoconference, 64 kbit/sec)
 - JPEG (Still pictures)
 - MPEG-1 (Audio-video up to 1.5 Mbit/sec, CD-ROM)
 - MPEG-2 (Audio-video, TV, HDTV, etc)
 - H.263 (Videoconference, 64 kbit/sec)
 - MPEG-4 Video (synthetic, natural audio-video, multimedia and Internet)
 - JPEG-2000 (Still pictures)
 - H.264/MPEG-4 AVC (Advanced Video Coding)
 - MPEG-4 AVC/SVC profile (Scalable Video Coding)
 - MPEG HEVC (High Efficiency Video Coding)
-