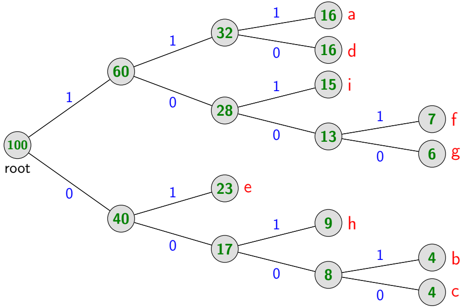
Lossless Coding III

a_k	p_k	$\boldsymbol{b}_k$
a	0.16	111
b	0.04	0001
c	0.04	0000
d	0.16	110
e	0.23	01
f	0.07	1001
g	0.06	1000
h	0.09	001
i	0.15	101



Random Processes with Memory

Example: Stationary Markov Process

- Statistical properties are given by conditional pmf $p(a_n|a_{n-1})$

a	$p(a\|a_0)$	$p(a\|a_1)$	$p(a\|a_2)$	$p(a)$	Huffman code
a_0	0.90	0.15	0.05	29/45	1
a_1	0.05	0.80	0.05	11/45	01
a_2	0.05	0.05	0.60	1/9	00

- Average codeword length for conventional Huffman code

$$\bar{\ell}_{\mathrm{SH}} = 61/45 \approx 1.3556$$
$$H(S) \approx 1.2575$$

- Can we exploit the dependencies between successive symbols?
  - ➡ Design a Huffman code for each condition
  - ➡ Switch code table after each symbol

Conditional Variable-Length Codes

Example: Stationary Markov Process

- Conditional code

	$S_{n-1} = a_0$		$S_{n-1} = a_1$		$S_{n-1} = a_2$	
a_k	p_k	code	p_k	code	p_k	code
a_0	0.90	1	0.15	01	0.05	01
a_1	0.05	01	0.80	1	0.05	00
a_2	0.05	00	0.05	00	0.60	1
	$\bar{\ell}_0 = 1.1$		$\bar{\ell}_1 = 1.2$		$\bar{\ell}_2 = 1.4$	

$$p(a_0) = 29/45$$
$$p(a_1) = 11/45$$
$$p(a_2) = 1/9$$

- Average codeword length for conditional code

$$\bar{\ell} = \sum_{\forall i} p(a_i) \cdot \bar{\ell}_i = \frac{521}{450} \approx 1.1578$$

$$\bar{\ell} < \bar{\ell}_{\mathrm{SH}} \approx 1.3556$$

$$\bar{\ell} < H(S) \approx 1.2575$$

Average Codeword Length for Conditional Codes

Bounds on Minimum Average Codeword Length

- For each condition $\{S_{n-1} = a_i\}$, we have the same relationship as in conventional variable-length coding

$$H(S_n \,|\, a_i) \leq \bar{\ell}_i < H(S_n \,|\, a_i) + 1 \tag{1}$$

where $H(S_n \,|\, a_i)$ is the conditional entropy given the event $\{S_{n-1} = a_i\}$,

$$H(S_n \,|\, a_i) = H(S_n \,|\, S_{n-1} = a_i) = -\sum_{\forall k} p(a_k \,|\, a_i) \log_2 p(a_k \,|\, a_i) \tag{2}$$

- Each condition $\{S_{n-1} = a_i\}$ occurs with the probability $p_i = \mathrm{P}(S_{n-1} = a_i)$
- Hence, the resulting bounds on $\bar{\ell}$ are given by

$$\left(\sum_{\forall i} p_i \, H(S_n \,|\, a_i) \right) \leq \underbrace{\left(\sum_{\forall i} p_i \, \bar{\ell}_i \right)}_{\bar{\ell}} < \left(\sum_{\forall i} p_i \, H(S_n \,|\, a_i) \right) + 1 \tag{3}$$

Conditional Entropy

Lower bound for conditional coding

- Conditional entropy of S_n given S_{n-1}

$$
\begin{aligned}
H(S_n \,|\, S_{n-1}) &= \sum_{\forall i} p(a_i)\, H(S_n \,|\, a_i) \\
&= \sum_{\forall i} p(a_i) \left(-\sum_{\forall k} p(a_k \,|\, a_i) \, \log_2 p(a_k \,|\, a_i) \right) \\
&= -\sum_{\forall i,k} p(a_k, a_i) \, \log_2 p(a_k \,|\, a_i) \qquad (4) \\
&= \mathrm{E}\Big\{ -\log_2 p(S_n \,|\, S_{n-1}) \Big\} \qquad (5)
\end{aligned}
$$

- Minimum average codeword length of a conditional code is bounded by

$$H(S_n \,|\, S_{n-1}) \leq \bar{\ell}_{\min} < H(S_n \,|\, S_{n-1}) + 1 \qquad (6)$$

General Conditional Coding

Arbitrary Condition

- Can use arbitrary random variable X as condition
- Example: Any function of already coded symbols $X = f(S_{n-1}, S_{n-2}, \cdots)$
- Design a code (codeword table) for each possible value of X

Conditional Entropy

- Conditional entropy of a random variable S given a random variable X

$$H(S\,|\,X) = \mathrm{E}\Big\{ -\log_2 p_{S|X}(S\,|\,X) \Big\} = -\sum_{\forall s,x} p_{SX}(s,x)\,\log_2 p_{S|X}(s\,|\,x) \qquad (7)$$

Bounds on Minimum Average Codeword Length

- Minimum average codeword length for conditional coding is bounded by

$$H(S\,|\,X) \le \bar{\ell}_{S|X} < H(S\,|\,X) + 1 \qquad (8)$$

Conditioning Does Not Increase Entropy

Conditional Entropy vs Marginal Entropy

- By using the divergence inequality, we obtain

$$
\begin{aligned}
H(S \,|\, X) &= -\sum_{\forall s,x} p_{SX}(s,x) \, \log_2 p_{S|X}(s \,|\, x) \quad \text{with} \quad p_{S|X}(s \,|\, x) = \frac{p_{SX}(s,x)}{p_X(x)} \\
&= -\sum_{\forall s,x} p_{SX}(s,x) \, \log_2 \left(\frac{p_{SX}(s,x)\, p_S(s)}{p_X(x)\, p_S(s)} \right) \\
&= -\sum_{\forall s} p_S(s) \, \log_2 p_S(s) - \sum_{\forall s,x} p_{SX}(s,x) \, \log_2 \left(\frac{p_{SX}(s,x)}{p_X(x)\, p_S(s)} \right) \\
&= H(S) - D(\, p_{SX} \,||\, p_S \, p_X \,) \qquad (9) \\
&\leq H(S) \qquad (10)
\end{aligned}
$$

- Equality is only obtained if the random variables S and X are independent

$$
D(\, p_{SX} \,||\, p_S \, p_X \,) = 0 \quad \Longleftrightarrow \quad p_{SX}(s,x) = p_S(s) \cdot p_X(x)
$$

Example: Stationary Markov Process

Summary on Markov Process Example

	conditional coding						conventional coding	
	$S_{n-1} = a_0$		$S_{n-1} = a_1$		$S_{n-1} = a_2$			
a_k	p_k	code	p_k	code	p_k	code	p_k	code
a_0	0.90	1	0.15	01	0.05	01	0.6444	1
a_1	0.05	01	0.80	1	0.05	00	0.2444	01
a_2	0.05	00	0.05	00	0.60	1	0.1111	00
	$H_0 = 0.5690$, $\bar{\ell}_0 = 1.1$		$H_1 = 0.8842$, $\bar{\ell}_1 = 1.2$		$H_2 = 1.3527$, $\bar{\ell}_2 = 1.4$			
	$H(S_n \mid S_{n-1}) = 0.7331$, $\bar{\ell}_{\mathrm{cond}} = 1.1578$						$H(S) = 1.2575$, $\bar{\ell}_{\mathrm{conv}} = 1.3556$	

➡ Conditioning reduces entropy from 1.2575 to 0.7331

➡ Conditioning reduces average codeword length from 1.3556 to 1.1578

Example for Conditional Code: H.264 | MPEG-4 AVC

Table 9-5 – coeff_token mapping to TotalCoeff(coeff_token) and TrailingOnes(coeff_token)

TrailingOnes (coeff_token)	TotalCoeff (coeff_token)	0 <= nC < 2	2 <= nC < 4	4 <= nC < 8	8 <= nC	nC = = −1	nC = = −2
0	0	1	11	1111	0000 11	01	1
0	1	0001 01	0010 11	0011 11	0000 00	0001 11	0001 111
1	1	01	10	1110	0000 01	1	01
0	2	0000 0111	0001 11	0010 11	0001 00	0001 00	0001 110
1	2	0001 00	0011 1	0111 1	0001 01	0001 10	0001 101
2	2	001	011	1101	0001 10	001	001
0	3	0000 0011 1	0000 111	0010 00	0010 00	0000 11	0000 0011 1
1	3	0000 0110	0010 10	0110 0	0010 01	0000 011	0001 100
2	3	0000 101	0010 01	0111 0	0010 10	0000 010	0001 011
3	3	0001 1	0101	1100	0010 11	0001 01	0000 1
0	4	0000 0001 11	0000 0111	0001 111	0011 00	0000 10	0000 0011 0
1	4	0000 0011 0	0001 10	0101 0	0011 01	0000 0011	0000 0010 1

(*continued*)

Example: Binary Markov Process

Black and White Document Scan

- Binary random process $\boldsymbol{S} = \{S\}$:

$$\begin{aligned} S = 0 \quad &\rightarrow \quad \text{white sample} \\ S = 1 \quad &\rightarrow \quad \text{black sample} \end{aligned}$$

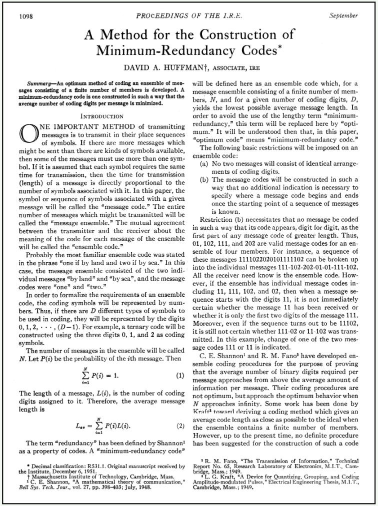
- Statistics measured over a large set of examples documents

$$\begin{aligned} p(0) &= 0.8 \\ p(0|0) &= 0.9 \end{aligned}$$

➡ Model: Stationary Markov process

➡ Determine remaining probabilities

$$\begin{aligned} p(1) &= 0.2 = 1 - p(0) \\ p(1|0) &= 0.1 = 1 - p(0|0) \\ p(0|1) &= 0.4 = p(1|0)\, p(0)/p(1) \\ p(1|1) &= 0.6 = 1 - p(0|1) \end{aligned}$$



1098 PROCEEDINGS OF THE I.R.E. September

A Method for the Construction of Minimum-Redundancy Codes*

DAVID A. HUFFMAN†, ASSOCIATE, IRE

Summary—An optimum method of coding an ensemble of messages consisting of a finite number of members is developed. A minimum-redundancy code is one constructed in such a way that the average number of coding digits per message is minimized.

INTRODUCTION

ONE IMPORTANT METHOD of transmitting messages is to transmit in their place sequences of symbols. If there are more messages which might be sent than there are kinds of symbols available, then some of the messages must use more than one symbol. If it is assumed that each symbol requires the same time for transmission, then the time for transmission (length) of a message is directly proportional to the number of symbols associated with it. In this paper, the symbol or sequence of symbols associated with a given message will be called the "message code." The entire number of messages which might be transmitted will be called the "message ensemble." The mutual agreement between the transmitter and the receiver about the meaning of the code for each message of the ensemble will be called the "ensemble code."

Probably the most familiar ensemble code was stated in the phrase "one if by land and two if by sea." In this case, the message ensemble consisted of the two individual messages "by land" and "by sea", and the message codes were "one" and "two."

In order to formalize the requirements of an ensemble code, the coding symbols will be represented by numbers. Thus, if there are D different types of symbols to be used in coding, they will be represented by the digits $0, 1, 2, \cdots, (D-1)$. For example, a ternary code will be constructed using the three digits 0, 1, and 2 as coding symbols.

The number of messages in the ensemble will be called N. Let $P(i)$ be the probability of the ith message. Then

$$\sum_{i=1}^{N} P(i) = 1. \quad (1)$$

The length of a message, $L(i)$, is the number of coding digits assigned to it. Therefore, the average message length is

$$L_{av} = \sum_{i=1}^{N} P(i)L(i). \quad (2)$$

The term "redundancy" has been defined by Shannon[1] as a property of codes. A "minimum-redundancy code" will be defined here as an ensemble code which, for a message ensemble consisting of a finite number of members, N, and for a given number of coding digits, D, yields the lowest possible average message length. In order to avoid the use of the lengthy term "minimum-redundancy," this term will be replaced here by "optimum." It will be understood then that, in this paper, "optimum code" means "minimum-redundancy code."

The following basic restrictions will be imposed on an ensemble code:

(a) No two messages will consist of identical arrangements of coding digits.

(b) The message codes will be constructed in such a way that no additional indication is necessary to specify where a message code begins and ends once the starting point of a sequence of messages is known.

Restriction (b) necessitates that no message be coded in such a way that its code appears, digit for digit, as the first part of any message code of greater length. Thus, 01, 102, 111, and 202 are valid message codes for an ensemble of four members. For instance, a sequence of these messages 1111022020101111102 can be broken up into the individual messages 111-102-202-01-01-111-102. All the receiver need know is the ensemble code. However, if the ensemble has individual message codes including 11, 111, 102, and 02, then when a message sequence starts with the digits 11, it is not immediately certain whether the message 11 has been received or whether it is only the first two digits of the message 111. Moreover, even if the sequence turns out to be 11102, it is still not certain whether 111-02 or 11-102 was transmitted. In this example, change of one of the two message codes 111 or 11 is indicated.

C. E. Shannon[1] and R. M. Fano[2] have developed ensemble coding procedures for the purpose of proving that the average number of binary digits required per message approaches from above the average amount of information per message. Their coding procedures are not optimum, but approach the optimum behavior when N approaches infinity. Some work has been done by Kraft[3] toward deriving a coding method which gives an average code length as close as possible to the ideal when the ensemble contains a finite number of members. However, up to the present time, no definite procedure has been suggested for the construction of such a code

* Decimal classification: R531.1. Original manuscript received by the Institute, December 6, 1951.
† Massachusetts Institute of Technology, Cambridge, Mass.
[1] C. E. Shannon, "A mathematical theory of communication," *Bell Sys. Tech. Jour.*, vol. 27, pp. 398–403; July, 1948.
[2] R. M. Fano, "The Transmission of Information," Technical Report No. 65, Research Laboratory of Electronics, M.I.T., Cambridge, Mass.; 1949.
[3] L. G. Kraft, "A Device for Quantizing, Grouping, and Coding Amplitude-modulated Pulses," Electrical Engineering Thesis, M.I.T., Cambridge, Mass.; 1949.

Example: Conventional and Conditional Coding

Black and White Document Scan (continued)

conditional coding

	$S_{n-1} = 0$		$S_{n-1} = 1$	
a_k	p_k	code	p_k	code
0	0.9	0	0.4	0
1	0.1	1	0.6	1
	$H_0 = 0.4690$, $\bar{\ell}_0 = 1$		$H_1 = 0.9710$, $\bar{\ell}_1 = 1$	
	$H(S_n \mid S_{n-1}) = 0.5694$, $\bar{\ell}_{\mathrm{cond}} = 1$			

conventional coding

p_k	code
0.8	0
0.2	1
$H(S) = 0.7219$, $\bar{\ell}_{\mathrm{conv}} = 1$	

- Conditioning does not improve coding efficiency in our case
- No codeword can be shorter than one bit, hence $\bar{\ell} \geq 1$

➡ Problem for sources with probability masses $\gg 0.5$

- **How can we increase coding efficiency?**

Variable-Length Coding For Blocks of Symbols

Block Codes

- Design variable length code for blocks of $N > 2$ symbols
- The N symbols of a block are jointly coded
- Optimal block code: Huffman algorithm for N-dimensional joint pmf

$$p(s_0, s_1, \cdots, s_{N-1}) = \mathrm{P}(S_0 = s_0, S_1 = s_1, \cdots, S_{N-1} = s_{N-1})$$

Block Huffman Coding for Black and White Document Scans

$N = 2$ symbols

$s_0 s_1$	$p(s_0, s_1)$	codewords
00	0.72	1
01	0.08	010
10	0.08	011
11	0.12	00

$$\bar{\ell}_2 = 1.44$$
$$\bar{\ell} = \bar{\ell}_2 / 2 = 0.72$$

$N = 3$ symbols $\longrightarrow$ $\bar{\ell} = 0.65$

$s_0 s_1 s_2$	$p(s_0, s_1, s_2)$	codewords
000	0.648	1
001	0.072	000
010	0.032	01000
011	0.048	0101
100	0.072	001
101	0.008	01001
110	0.048	0110
111	0.072	0111

Block Entropy

Bounds on minimum average codeword length

- Let $\bar{\ell}_N$ denote the average codeword length for joint coding of N symbols
- For combined alphabet: Same relationship as in scalar case

➡ Minimum average codeword length is bounded by

$$\mathrm{E}\{-\log_2 p(S_0,\cdots,S_{N-1})\} \leq \bar{\ell}_N < \mathrm{E}\{-\log_2 p(S_0,\cdots,S_{N-1})\}+1 \tag{11}$$

Block Entropy

- Lower bound for average codeword length for N symbols

$$\begin{aligned} H_N(\boldsymbol{S}) &= H(S_0,S_1,\cdots,S_{N-1}) \\ &= \mathrm{E}\{-\log_2 p(S_0,\cdots,S_{N-1})\} \\ &= -\sum_{a_0,a_1,\cdots,a_{N-1}} p(a_0,a_1,\cdots,a_{N-1}) \log_2 p(a_0,a_1,\cdots,a_{N-1}) \end{aligned} \tag{12}$$

Bounds on Minimum Average Codeword Length

Variable Length Coding of Fixed-Length Symbol Blocks

- Bounds for minimum average codeword length per N symbols

$$H_N(\boldsymbol{S}) \leq \bar{\ell}_N < H_N(\boldsymbol{S}) + 1 \tag{13}$$

- Bounds for minimum average codeword length per symbol

$$\frac{H_N(\boldsymbol{S})}{N} \leq \bar{\ell} < \frac{H_N(\boldsymbol{S})}{N} + \frac{1}{N} \tag{14}$$

Chain Rule for Entropies

- Remember: Conditional probabilities

$$p(s_0, s_1, \cdots, s_{N-1}) = p(s_0)\, p(s_1 \,|\, s_0)\, p(s_2 \,|\, s_0, s_1) \cdots p(s_{N-1} \,|\, s_0, \cdots, s_{N-2})$$

- Consequence: Chain rule for entropies

$$\begin{aligned} H(S_0, S_1, \cdots, S_{N-1}) \;=\; & H(S_0) + H(S_1 \,|\, S_0) + H(S_2 \,|\, S_0, S_1) + \cdots \\ & \cdots + H(S_{N-1} \,|\, S_0, \cdots, S_{N-2}) \end{aligned} \tag{15}$$

Properties of Block Entropy

Block Entropy vs Conditional Entropy

- Remember: Chain rule

$$H(S_0, S_1, \cdots, S_{N-1}) = H(S_0) + H(S_1 \,|\, S_0) + H(S_2 \,|\, S_0, S_1) + \cdots \\ \cdots + H(S_{N-1} \,|\, S_0, \cdots, S_{N-2})$$

- Remember: Conditioning never increases entropy

$$H(S_0, S_1, \cdots, S_{N-1}) \geq N \cdot H(S_{N-1} \,|\, S_0, \cdots, S_{N-2}) \tag{16}$$

Block Entropy for Different Block Sizes

- Remember: Conditional probabilities

$$p(s_0, s_1, \cdots, s_{N-1}) = p(s_0, \cdots, s_{N-2})\, p(s_{N-1} \,|\, s_0, \cdots, s_{N-2})$$

- Consequence for block entropy

$$H(S_0, \cdots, S_{N-1}) = H(S_0, \cdots, S_{N-2}) + H(S_{N-1} \,|\, S_0, \cdots, S_{N-2}) \tag{17}$$

Increasing Block Size Never Increases Lower Bound

Effect of Increasing Block Size

- For stationary random processes, we have shown

$$
\begin{aligned}
H_N(\boldsymbol{S}) \; &= \; H_{N-1}(\boldsymbol{S}) + H(S_n \,|\, S_{n-1}, \cdots, S_{n-N+1}) \\
H_N(\boldsymbol{S}) \; &\geq \; N \cdot H(S_n \,|\, S_{n-1}, \cdots, S_{n-N+1})
\end{aligned}
$$

- Combining these relationships yields

$$
\begin{aligned}
N \cdot H_N(\boldsymbol{S}) \; &\leq \; N \cdot H_{N-1}(\boldsymbol{S}) + H_N(\boldsymbol{S}) \\
(N-1) \cdot H_N(\boldsymbol{S}) \; &\leq \; N \cdot H_{N-1}
\end{aligned}
$$

➡ Hence: **Increasing block size never increases lower bound**

$$
\frac{H_N(\boldsymbol{S})}{N} \leq \frac{H_{N-1}(\boldsymbol{S})}{N-1} \tag{18}
$$

➡ Equality if $H(S_n \,|\, S_{n-1}, \cdots) = H(S_n)$ (iid processes)

Fundamental Lossless Source Coding Theorem

Entropy Rate

- Definition of **entropy rate**

$$\bar{H}(\boldsymbol{S}) = \lim_{N \to \infty} \frac{H(S_0, \cdots, S_{N-1})}{N} = \lim_{N \to \infty} \frac{H_N(\boldsymbol{S})}{N} \tag{19}$$

- Limit (for $N \to \infty$) of lower bound for block coding
- The limit always exists for stationary random sources

Fundamental Lossless Source Coding Theorem

- Average codeword length for all lossless codes is bounded by

$$\bar{\ell} \geq \bar{H}(\boldsymbol{S}) = \lim_{N \to \infty} \frac{H_N(\boldsymbol{S})}{N} \tag{20}$$

- Asymptotically achievable with block Huffman coding for $N \to \infty$

➡ Size of codeword tables exponentially increases with N

Entropy Rate for Special Sources

IID Processes

- Use chain rule for entropies

$$
\begin{aligned}
\bar{H}(\boldsymbol{S}) &= \lim_{N\to\infty} \frac{1}{N} H(S_0, S_1, \cdots, S_{N-1}) \\
&= \lim_{N\to\infty} \frac{1}{N} \Big(H(S_0) + H(S_1 \,|\, S_0) + H(S_2 \,|\, S_0, S_1) + \cdots \Big) \\
&= \lim_{N\to\infty} \frac{1}{N} \sum_{n=0}^{N-1} H(S_n) \\
&= \lim_{N\to\infty} \frac{1}{N} \Big(N \cdot H(S) \Big) \\
&= H(S) \qquad (21)
\end{aligned}
$$

- Note: Block Huffman coding may still improve coding efficiency compared to scalar Huffman coding

Entropy Rate for Special Sources

Stationary Markov Processes

- Use chain rule for entropies

$$
\begin{aligned}
\bar{H}(\boldsymbol{S}) &= \lim_{N\to\infty} \frac{1}{N} H(S_0, S_1, \cdots, S_{N-1}) \\
&= \lim_{N\to\infty} \frac{1}{N} \Big(H(S_0) + H(S_1 \,|\, S_0) + H(S_2 \,|\, S_0, S_1) + \cdots \Big) \\
&= \lim_{N\to\infty} \frac{1}{N} \Big(H(S) + (N-1) \cdot H(S_n \,|\, S_{n-1}) \Big) \\
&= \lim_{N\to\infty} \frac{H(S)}{N} + \lim_{N\to\infty} \frac{N-1}{N} H(S_n \,|\, S_{n-1}) \\
&= H(S_n \,|\, S_{n-1}) \qquad (22)
\end{aligned}
$$

- Note: Can also use conditional block Huffman coding

➡ Switch block code based on conditional variable

Example: Stationary Markov Process

Markov Source

a	$p(a\|a_0)$	$p(a\|a_1)$	$p(a\|a_2)$
a_0	0.90	0.15	0.05
a_1	0.05	0.80	0.05
a_2	0.05	0.05	0.60

$$H(S) = 1.2575$$
$$\bar{H}(\boldsymbol{S}) = 0.7331$$

N	$\frac{H_N(\boldsymbol{S})}{N}$	$\bar{\ell} = \frac{\bar{\ell}_N}{N}$	number of codewords
1	1.2575	1.3556	3
2	0.9953	1.0094	9
3	0.9079	0.9150	27
4	0.8642	0.8690	81
5	0.8380	0.8462	243
6	0.8205	0.8299	729
7	0.8080	0.8153	2187
8	0.7987	0.8027	6561
9	0.7914	0.7940	19683

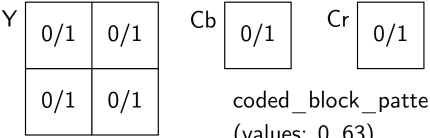
Scalar Huffman code:

$$\bar{\ell} = 1.3556$$

Conditional Huffman code:

$$\bar{\ell} = 1.1578$$

Example for Block Huffman Code: CBF in MPEG-2



coded_block_pattern = xxxxxx (bit mask)
(values: 0..63)

Table B.9 – Variable length codes for coded_block_pattern

coded_block_pattern VLC code	cbp	coded_block_pattern VLC code	cbp
111	60	0001 1100	35
1101	4	0001 1011	13
1100	8	0001 1010	49
1011	16	0001 1001	21
1010	32	0001 1000	41
1001 1	12	0001 0111	14
1001 0	48	0001 0110	50
1000 1	20	0001 0101	22
1000 0	40	0001 0100	42
0111 1	28	0001 0011	15
0111 0	44	0001 0010	51

(continued)

V2V Codes

Generalization of Block Codes

- Assign codewords to **symbol sequences of variable length**
- How to select symbol sequences?
  - ➡ All messages must be representable by symbol sequences
  - ➡ Desirable: Redundancy-free set of symbol sequences

Examples

- Consider binary symbol alphabet $\mathcal{A} = \{a, b\}$

code A	
aaaa	0
aaab	10
aab	110
bba	1110
ba	1111

abbbb... ?

code B	
aaa	000
aa	01
a	1
b	0010
bb	0011

redundant!

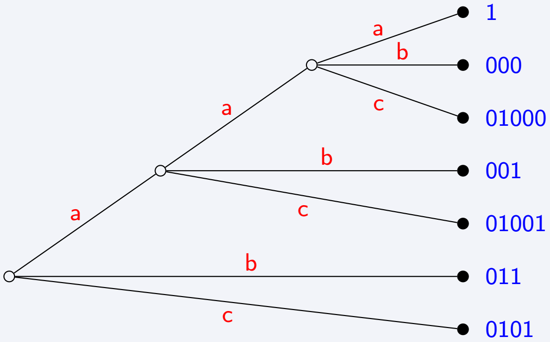
code C	
aaaa	0
aaab	10
aab	110
ab	1110
b	1111

suitable

Suitable Set of Variable-Length Symbol Sequences

Finite alphabet of M letters

- Select symbol sequences that are representable by **full M-ary tree**



➡ All messages are representable by a concatenation of symbol sequences

➡ Redundancy-free set of symbol sequences

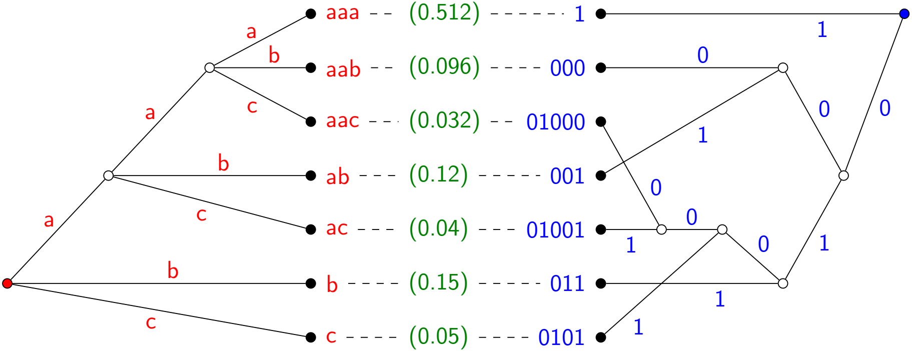
➡ Instantaneous encodable codes

V2V Codes as Double Tree

IID Source

symbol	probability
a	0.80
b	0.15
c	0.05

entropy rate:	$\bar{H}(\boldsymbol{S}) = 0.88418$	(marginal entropy)
scalar Huffman:	$\bar{\ell} = 1.2$	(3 codewords)
2-symbol blocks:	$\bar{\ell} = 0.93375$	(9 codewords)
V2V code:	$\bar{\ell} = 0.88934$	(7 codewords)
redundancy:	$\varrho = 0.00516$	(0.58%)



Average Codeword Length

V2V Code Design

- Choose set of symbol sequences that are representable by full M-ary tree
- Determine pmf of symbol sequences (leaf nodes of the M-ary tree)
- Design Huffman code for pmf of variable-length symbol sequences

Average Codeword Length of V2V Codes

- Given the pmf for the leaf nodes, the average codeword length is given by

$$\bar{\ell} = \frac{\sum_{k=0}^{L-1} p_k \, \ell_k}{\sum_{k=0}^{L-1} p_k \, n_k} = \frac{\textit{average codeword length per sequence}}{\textit{average number of symbols per sequence}} \tag{23}$$

with
L : number of symbols sequences (leaf nodes)
p_k : probability of k-th symbol sequence
ℓ_k : length of codeword assigned to k-th symbol sequence
n_k : number of symbols in k-th symbol sequence

PMF for Symbol Sequences

How to determine pmf for variable-length sequences?

- In general: Probability that a new symbol sequence starts depends on previous symbols in message

➡ Probability of a symbol sequence $\boldsymbol{a} = (a_0, a_1, \cdots, a_{K-1})$

$$p(\boldsymbol{a}) = p(a_0 \,|\, \mathcal{B})\, p(a_1 \,|\, a_0, \mathcal{B}) \cdots p(a_{K-1} \,|\, a_0, \cdots, a_{K-2}, \mathcal{B}) \qquad (24)$$

with $\mathcal{B}$ being the event that the preceding message symbols were coded using a complete symbol sequence of the given set (V2V code tree)

➡ Conditional pmfs $p(a_m \,|\, a_0, \cdots, a_{m-1}, \mathcal{B})$ are given by

- Conditional pmfs $p(a_m \,|\, a_0, \cdots, a_{m-1})$ of the random process
- Structure of the V2V code (M-ary symbol tree)

IID Processes

- No dependencies on previous symbols

$$p(\boldsymbol{a}) = p(a_0)\, p(a_1) \cdots p(a_{K-1}) \qquad (25)$$

PMF for Symbol Sequences

Markov Processes

- Probability of a symbol sequence $\boldsymbol{a} = (a_0, a_1, \cdots, a_{K-1})$ is given by

$$p(\boldsymbol{a}) = p(a_0 \,|\, \mathcal{B})\, p(a_1 \,|\, a_0)\, p(a_2 \,|\, a_1) \cdots p(a_{K-1} \,|\, a_{K-2}) \tag{26}$$

- Probability that a new symbol sequence starts with letter a_m is given by

$$p(a_m \,|\, \mathcal{B}) = \sum_{k=0}^{L-1} p(a_m \,|\, a^k_{n_k-1}) \cdot p(a^k_{n_k-1} \,|\, a^k_{n_k-2}) \cdot \ldots \cdot p(a^k_1 \,|\, a^k_0) \cdot p(a^k_0 \,|\, \mathcal{B}) \tag{27}$$

with L : number of symbols sequences
n_k : number of symbols in k-th symbol sequence
a^k_n : n-th symbol in k-th symbol sequence

- Together with the condition $\sum_m p(a_m \,|\, \mathcal{B}) = 1$ we have a linear equation system with a unique solution

➡ Can calculate all unknown probabilities $p(a_n \,|\, \mathcal{B})$ and thus also the pmf for the symbol sequences (leaf nodes)

Example for Binary Markov Source

Black and White Document Scan

conditional & marginal pmf:

a	p(a|0)	p(a|1)	p(a)
0	0.9	0.4	0.8
1	0.1	0.6	0.2

sequences	
00000	$p(0\|\mathcal{B}) = p(0\|\mathcal{B}) \cdot p(0\|0) \cdot p(0\|0) \cdot p(0\|0) \cdot p(0\|0) \cdot p(0\|0) +$
00001	$p(0\|\mathcal{B}) \cdot p(0\|0) \cdot p(0\|0) \cdot p(0\|0) \cdot p(1\|0) \cdot p(0\|1) +$
0001	$p(0\|\mathcal{B}) \cdot p(0\|0) \cdot p(0\|0) \cdot p(1\|0) \cdot p(0\|1) +$
001	$p(0\|\mathcal{B}) \cdot p(0\|0) \cdot p(1\|0) \cdot p(0\|1) +$
01	$p(0\|\mathcal{B}) \cdot p(1\|0) \cdot p(0\|1) +$
10	$p(1\|\mathcal{B}) \cdot p(0\|1) \cdot p(0\|0) +$
110	$p(1\|\mathcal{B}) \cdot p(1\|1) \cdot p(0\|1) \cdot p(0\|0) +$
111	$p(1\|\mathcal{B}) \cdot p(1\|1) \cdot p(1\|1) \cdot p(0\|1) +$

$$
\begin{aligned}
p(0|\mathcal{B}) &= 0.72805 \cdot p(0|\mathcal{B}) + 0.72 \cdot p(1|\mathcal{B}) \\
&= 0.72805 \cdot p(0|\mathcal{B}) + 0.72 \cdot (1 - p(0|\mathcal{B}))
\end{aligned}
$$

$$\Longrightarrow p(0|\mathcal{B}) = 0.725842 \qquad \Longrightarrow p(1|\mathcal{B}) = 0.274157$$

Example for Binary Markov Source

Black and White Document Scan (continued)

- Calculate pmf for symbol sequences and design Huffman code

$$p(a_0, a_1, \cdots, a_{k-1}) = p(a_0|\mathcal{B})\, p(a_1|a_0) \cdots p(a_{k-1}|a_{k-2})$$

sequences	probabilities	Huffman code
00000	0.476226	1
00001	0.052914	0100
0001	0.058793	0101
001	0.065326	0110
01	0.072584	0111
10	0.109663	001
110	0.065798	0000
111	0.098697	0001

$$\bar{H}(\boldsymbol{S}) = 0.59049$$

$$\bar{\ell} = 0.62561$$

$$\varrho = 0.03512 \quad (5.9\%)$$

➡ More efficient than block Huffman code with the same number of codewords

$$\bar{\ell} = 0.65 \quad \Longrightarrow \quad \varrho = 0.05951 \quad (10.1\%)$$

Optimal V2V Codes?

Optimization Problem

- Best V2V code for given maximum number of codewords?

➡ No known design algorithm

➡ Exhaustive search over all possible symbol trees

Exhaustive V2V code design

- Investigate all symbol trees with a maximum number of leaf nodes
  - Create symbol tree
  - Calculate probabilities for leaf nodes (symbol sequences)
  - Determine codewords using Huffman algorithm
  - Calculate average codeword length
- Choose symbol tree (and code) that minimizes average codeword length

➡ Extremely complex

Example for Optimal V2V codes

Markov Source

a	$p(a\|a_0)$	$p(a\|a_1)$	$p(a\|a_2)$
a_0	0.90	0.15	0.05
a_1	0.05	0.80	0.05
a_2	0.05	0.05	0.60

$$H(S) = 1.2575$$
$$\bar{H}(\boldsymbol{S}) = 0.7331$$

V2V:

N_C	$\bar{\ell}$
5	1.1784
7	1.0551
9	1.0049
11	0.9733
13	0.9412
15	0.9293
17	0.9074
19	0.8980
21	**0.8891**

(N_C: number of codewords)

Scalar Huffman code:

$\bar{\ell} = 1.3556$

Conditional Huffman code:

$\bar{\ell} = 1.1578$

Block Huffman code:

$N_C = 9 \;:\; \bar{\ell} = 1.0094$

$N_C = 27 \;:\; \bar{\ell} = 0.9150$

V2V Codes In Practice

In Practice

- Typically: Only structured V2V codes

→ Set of symbol sequences follows a certain structure

Well-Known Example: Run-Level Coding

- Often: Long sequences of symbols equal to zero
- Map sequence a symbols (transform coefficients) into (run,level) pairs, including a special end-of-block (eob) symbol

  level : value of next non-zero symbol
  run : number of zero symbols that precede next non-zero symbol
  eob : all following symbols are equal to zero (end-of-block)

→ Assign codewords to (run,level) pairs (including eob symbol)

- **Example**:

  64 symbols: 5 3 0 0 0 1 0 1 0 0 1 0 0 0 0 0 0 0 0 ...
  (run,level) pairs: (0,5) (0,3) (3,1) (1,1) (2,1) (eob)

Run-Level Coding in MPEG-2 Video

Table B.14 – DCT coefficients Table zero

Variable length code (Note 1)	Run	Level
10 (Note 2)	End of Block	
1 s (Note 3)	0	1
11 s (Note 4)	0	1
011 s	1	1
0100 s	0	2
0101 s	2	1
0010 1 s	0	3
0011 1 s	3	1
0011 0 s	4	1
0001 10 s	1	2
0001 11 s	5	1
0001 01 s	6	1
0001 00 s	7	1

(*continued*)

Adaptive VLC codes

In Practice

- Messages with different properties
  - Classical music has other properties than heavy metal
  - MRT images have other properties than natural images
  - Cartoon movies have other properties than documentaries
- Messages itself have typically instationary statistical properties

➜ Optimize code tables for actual data

Adaptation of Variable-Length Codes

Two basic approaches

1. **Forward adaptation**
   ➜ Transmit adaptation signal (pmf, code, ...) as side information
2. **Backward adaptation**
   ➜ Adapt code simultaneously at encoder and decoder

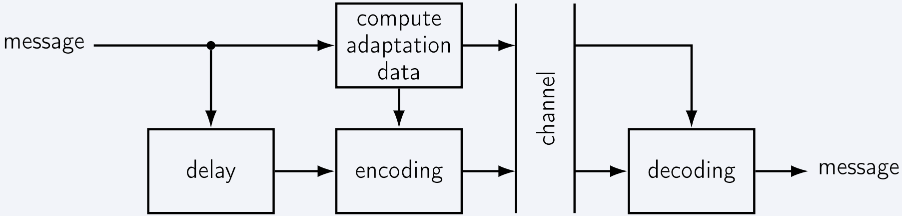
- Possible to combine forward and backward adaptation

Forward Adaptation

Principle

- Gather statistics for large enough block of source symbols
- Transmit adaptation signal to decoder (e.g., at start of message)
  - Probability mass function
  - Codeword table
- Disadvantage: Increased bit rate due to side information

➜ Example: JPEG: Transmission of codeword table

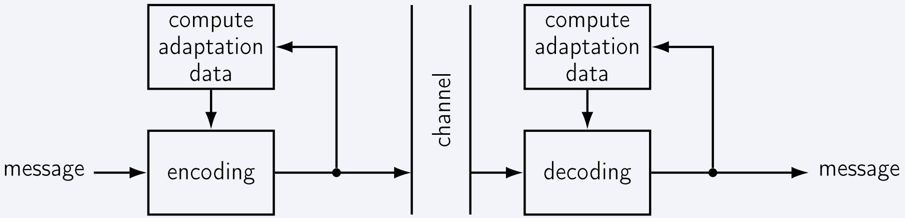


Backward Adaptation

Principle

- Gather statistics at both encoder and decoder
- Adapt code simultaneously at encoder and decoder during coding
  - Probability mass functions
  - Codeword tables
- Disadvantage: Decreased error robustness

➜ Example: H.264/AVC and H.265/HEVC: Adaptive arithmetic coding



Shannon-Fano-Elias Coding and Arithmetic Coding

Variable-Length Codes

- Optimal code for given pmf: Huffman code
- Discussed: Scalar codes, conditional codes, block codes, V2V codes

➡ Scalar and conditional codes can be very inefficient

➡ Entropy rate can be asymptotically achieved using block codes ($N \to \infty$)

➡ Impractical: Codeword tables grow exponentially with N

Practical Block Codes

- **Shannon-Fano-Elias codes**
  - ➡ Sub-optimal block codes (still close to optimal for large N)
  - ➡ Iterative construction of codewords (no need to store codeword table)
- **Arithmetic codes**
  - ➡ Fixed-precision variant of Shannon-Fano-Elias codes
  - ➡ State-of-the art in lossless coding (very flexible)

Shannon-Fano-Elias Coding: Framework

Random Process Model

- Assume stationary discrete random process $\boldsymbol{S} = \{S_n\}$
- Statistical properties are given by N-th order joint pmf

$$p(\boldsymbol{s}) = \mathrm{P}(\boldsymbol{S} = \boldsymbol{s}) = \mathrm{P}(S_0 = s_0,\, S_1 = s_1, \cdots, S_{N-1} = s_{N-1}) \tag{28}$$

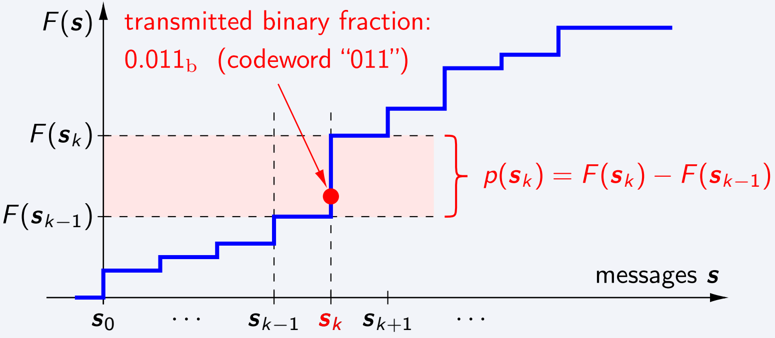
- Messages $\boldsymbol{s}^{(L)} = \{s_0, s_1, s_2, \cdots, s_{L-1}\}$ of L symbols are finite-length realizations of random process $\boldsymbol{S}$

Coding Framework

- Code blocks of N symbols (N is known to encoder and decoder)
- Consider two configurations:
  1. $N < L$: Message is split into multiple blocks of symbols
     ➡ Prefix code
  2. $N = L$: All symbols of a message are jointly coded
     ➡ No prefix code required

Shannon-Fano-Elias Coding

Basic Idea



- Order all symbol sequences with N symbols: $\boldsymbol{s}_0, \boldsymbol{s}_1, \boldsymbol{s}_2, \cdots$
- Each symbol sequence is associated with an interval of the cdf $F(\boldsymbol{s})$
- Transmit any number (as binary fraction) inside the corresponding interval
  - ➡ Required number of bits depends on probability of symbol sequence

Mapping of Symbol Sequences to Intervals

Order of Symbol Sequences

- Require a defined order of symbol sequences (with N symbols)
- Order $\{\boldsymbol{s}_0, \boldsymbol{s}_1, \boldsymbol{s}_2, \cdots\}$ must be known to encoder and decoder

Mapping to Intervals

- Each symbol sequence $\boldsymbol{s}_k$ is mapped to an **half-open interval** $\mathcal{I}(\boldsymbol{s}_k) \subset [0, 1)$

$$\mathcal{I}(\boldsymbol{s}_k) = \Big[L(\boldsymbol{s}_k),\ U(\boldsymbol{s}_k) \Big) = \Big[L(\boldsymbol{s}_k),\ L(\boldsymbol{s}_k) + W(\boldsymbol{s}_k) \Big) \tag{29}$$

- Intervals $\mathcal{I}(\boldsymbol{s}_k)$ can be characterized by
  - ➡ **lower interval boundary** $L(\boldsymbol{s}_k)$

$$L(\boldsymbol{s}_k) = F(\boldsymbol{s}_{k-1}) = \mathrm{P}(\boldsymbol{S} < \boldsymbol{s}_k) = \sum_{\forall i < k} p(\boldsymbol{s}_i) \tag{30}$$

  - ➡ **interval width** $W(\boldsymbol{s}_k)$

$$W(\boldsymbol{s}_k) = F(\boldsymbol{s}_k) - F(\boldsymbol{s}_{k-1}) = \mathrm{P}(\boldsymbol{S} = \boldsymbol{s}_k) = p(\boldsymbol{s}_k) \tag{31}$$

Unique Identification of Intervals

Disjoint Intervals

- Half-open intervals $\mathcal{I}(\boldsymbol{s}_k)$ are disjoint by definition
- All real numbers $v \in [0, 1)$ belong to exactly one interval

Representative Number Inside Interval

- Transmit any number $v \in \mathcal{I}(\boldsymbol{s}_k)$ for uniquely identifying
  - ➡ the interval $\mathcal{I}(\boldsymbol{s}_k)$ and, thus,
  - ➡ the symbol sequence $\boldsymbol{s}_k$
- Represent number $v \in \mathcal{I}(\boldsymbol{s}_k)$ as binary fraction with K bits of precision

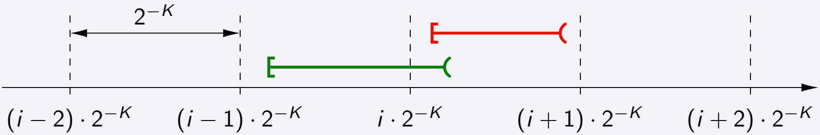
$$v = (0.b_0 b_1 b_2 \cdots b_{K-1})_{\mathrm{b}} = \sum_{i=0}^{K-1} b_i \cdot 2^{-(i+1)} \tag{32}$$

➡ Number v is an integer multiple of 2^{-K}

➡ Codeword is given by bit sequence $\boldsymbol{b} = \{b_0, b_1, b_2, \cdots, b_{K-1}\}$ of K bits

How Many Bits for Identifying an Interval ?

Required Number of Bits



- Distance between successive binary fractions of K bits is 2^{-K}

➡ For guaranteeing that a binary fraction of K bits falls inside an interval $\mathcal{I}(\boldsymbol{s}_k)$ of width $W(\boldsymbol{s}_k)$, we require

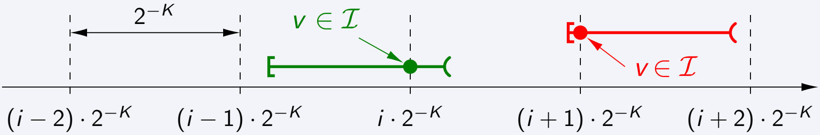
$$\begin{aligned} W(\boldsymbol{s}_k) &\geq 2^{-K} \\ K &\geq -\log_2 W(\boldsymbol{s}_k) \end{aligned} \tag{33}$$

➡ Hence, we choose

$$K = K(\boldsymbol{s}_k) = \left\lceil -\log_2 W(\boldsymbol{s}_k) \right\rceil = \left\lceil -\log_2 p(\boldsymbol{s}_k) \right\rceil \tag{34}$$

How to Select Interval Representative and Codeword ?

Interval Representative



- Round up lower interval boundary L to next binary fraction of K bits

➡ For interval $\mathcal{I} = [L, L+W)$, choose binary number v according to

$$v = \lceil L \cdot 2^K \rceil \cdot 2^{-K} \qquad \text{with} \qquad K = \lceil -\log_2 W \rceil \tag{35}$$

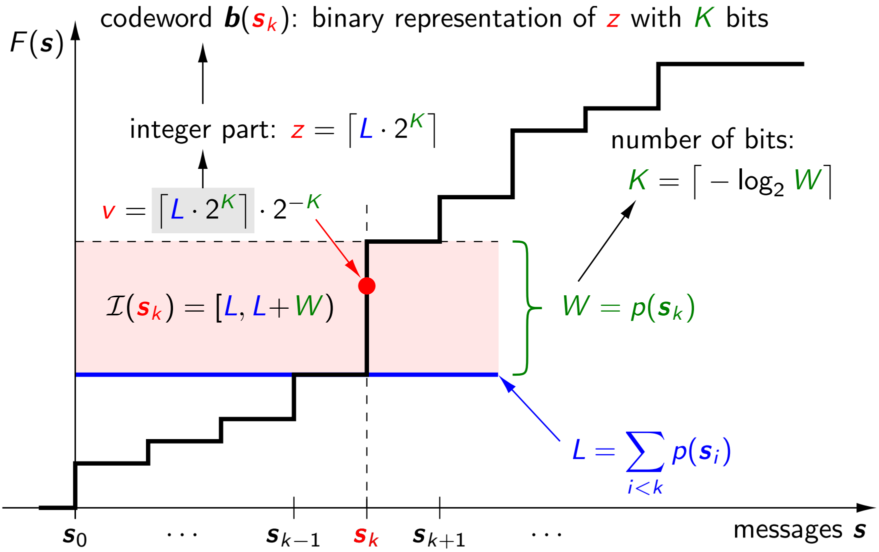
Codewords

- K fractional bits of interval representative $v = (0.b_0 b_1 b_2 \cdots b_{K-1})_{\mathrm{b}}$

➡ Binary representation $[b_0 b_1 \cdots b_{K-1}]$ with K bits of integer number

$$z = \lceil L \cdot 2^K \rceil = v \cdot 2^K \tag{36}$$

Shannon-Fano-Elias Encoding: Illustration



Shannon-Fano-Elias Encoding: Summary

Determination of Codewords

- Given: Ordered set of symbol sequences $\{\boldsymbol{s}_k\}$ with associated pmf $\{p_k\}$
- Construct codeword $\boldsymbol{b}_k = \boldsymbol{b}(\boldsymbol{s}_k)$ for any particular sequence $\boldsymbol{s}_k$ by

1. Determine interval width W_k and lower interval boundary L_k

$$W_k = p_k \tag{37}$$

$$L_k = \sum_{i<k} p_i \tag{38}$$

2. Determine codeword length K_k

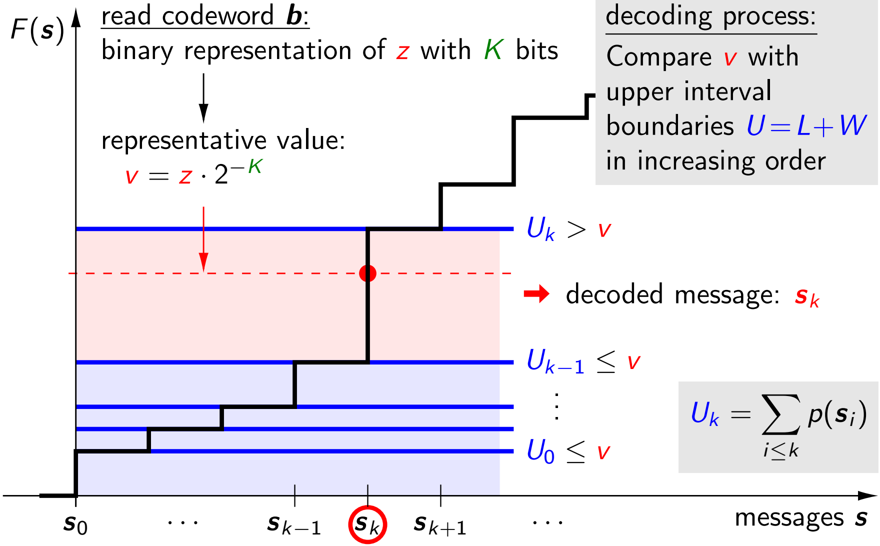
$$K_k = \lceil -\log_2 W_k \rceil \tag{39}$$

3. Determine representative integer z_k

$$z_k = \lceil L_k \cdot 2^{K_k} \rceil \tag{40}$$

4. Codeword $\boldsymbol{b}_k$: Binary representation of z_k with K_k bits

Shannon-Fano-Elias Decoding: Illustration



decoding process:

Compare v with upper interval boundaries $U = L + W$ in increasing order

$$U_k = \sum_{i \leq k} p(\boldsymbol{s}_i)$$

Shannon-Fano-Elias Decoding: Summary

Decoding of a Symbol Sequence

- Given: Ordered set of symbol sequences $\{\boldsymbol{s}_k\}$ with associated pmf $\{p_k\}$

1 Read codeword $\boldsymbol{b}$: Binary representation of **integer** z with K **bits**

2 Initialization of iterative decoding

- representative value: $v = z \cdot 2^{-K}$
- iteration index: $k = 0$
- upper interval boundary: $U_0 = L_0 + W_0 = p_0$

3 Compare v with U_k

- If $v < U_k$
  - ➡ Output decoded symbol sequence $\boldsymbol{s}_k$
  - ➡ Terminate decoding
- Otherwise $(v \geq U_k)$
  - ➡ Update iteration index: $k = k + 1$
  - ➡ Update upper interval boundary: $U_k = U_{k-1} + p_k$
  - ➡ Goto step **3**

Summary: Extended Variable-Length Coding

Conditional Codes

- Switching between codeword tables (depending on previous symbol(s))
- Bound for average codeword length: Conditional entropy
- Conditioning never increases average codeword length or entropy

Block Codes

- Variable-length code for blocks of N symbols
- Bound for average codeword length: Block entropy divided by N
- Increasing block size never increases average codeword length or entropy
- Fundamental lossless coding theorem: $\bar{\ell} \geq \bar{H}$ (entropy rate)

V2V Codes

- Assign codewords to variable-length symbol sequences
- Important example: Run-level coding of transform coefficients

Adaptive Variable-Length Codes

- Adapt code during encoding/decoding (forward/backward adaptation)

Intermediate Summary: Shannon-Fano-Elias Coding

Shannon-Fano-Elias Codes

- Special Block Code
- In general: Worse than Huffman code for same block size
- On-the-fly construction of codewords (no need to store codeword table)
- On-the-fly decoding of messages

Next

- **Iterative encoding and decoding** for Shannon-Fano-Elias codes
- **Arithmetic coding** as practical implementation of Shannon-Fano-Elias codes