Multimedia Systems

WS 2009/2010

Compression

Prof. Dr. Paul Müller
University of Kaiserslautern, Germany
Integrated Communication Systems Lab

Email: pmueller@informatik.uni-kl.de

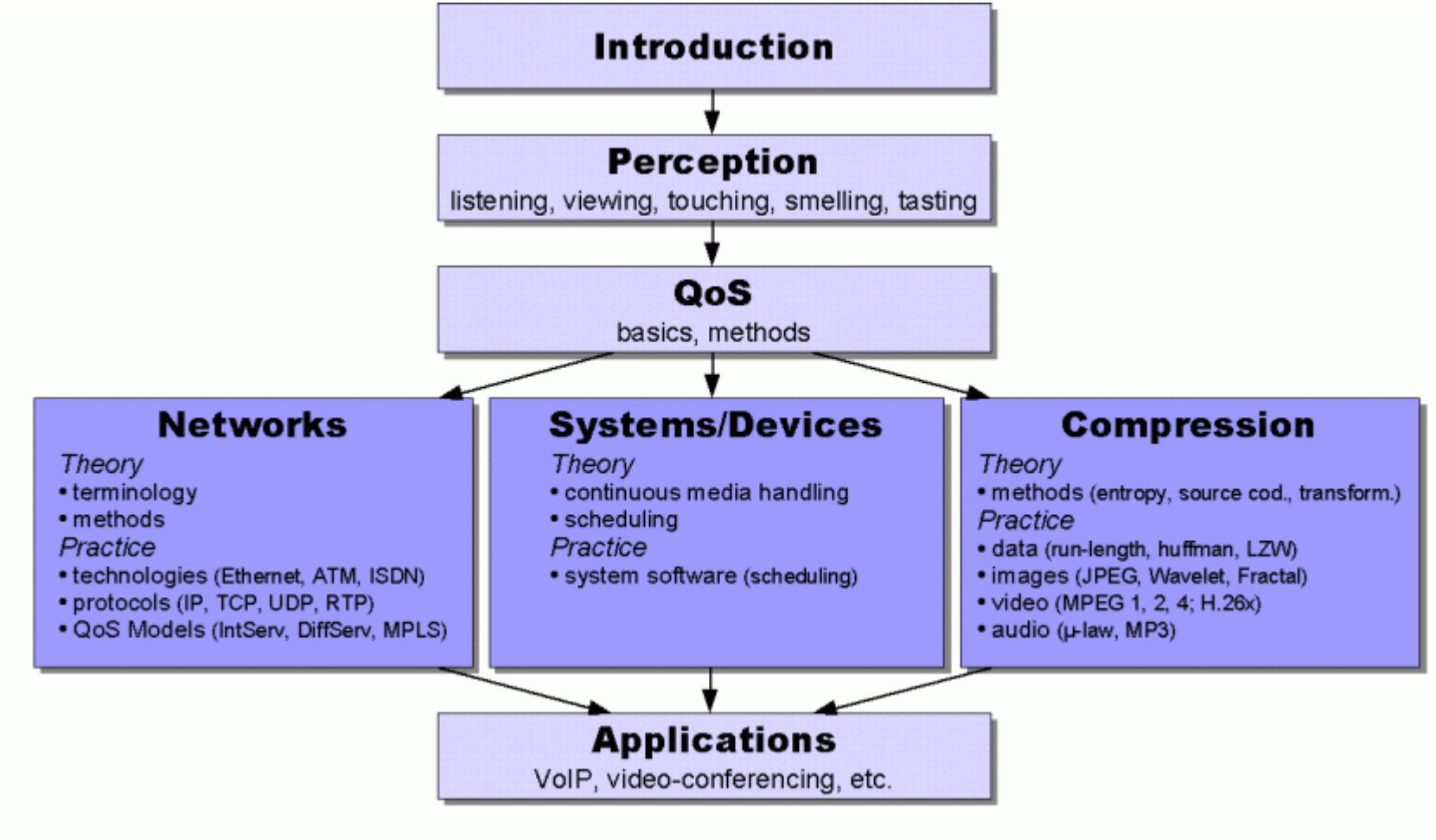
Literature

- Ralf Steinmetz
  Multimedia-Technologie. Grundlagen, Komponenten und Systeme.
  3. Auflage, Springer, 2000
- Andreas Holzinger
  Basiswissen Multimedia Band 1: Technik
  Vogel Buchverlag, 2001
- Konrad Froitzheim
  Multimedia Kommunikation
  dpunkt, 1997
- Grenville Armitage
  Quality of Service in IP Networks
  Macmillan Technical Publishing, 2000
- Zheng Wang
  Internet QoS - Architectures ans Mechanisms for Quality of Service
  Morgan Kaufmann Publishers, 2001

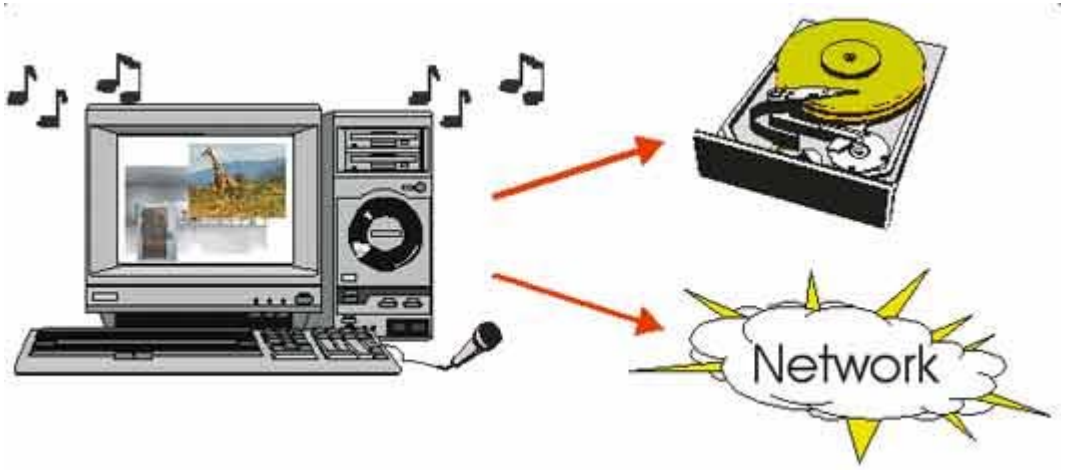
Literature (2)

- Tilo Strutz
  Bilddatenkompression, Grundlagen, Codierung, MPEG, JPEG Vieweg, 2000
- Khalid Sayood
  Introduction to Data Compression, second edition
  Morgan Kaufmann, 2000
- Gregory A. Baxes
  Digital Image Processing,
  Wiley, 1994
- Compression FAQ
  http://www.faqs.org/faqs/compression-faq/

Sitemap



5.1. Motivation (1)



- **e.g. video sequence**
  - 25 images/sec.
  - 3 Byte/Pixel
  - Image resolution 640 * 480 Pixel
- **Data rate = 640 * 480 * 3 Byte * 25/s = 23040000 Byte/s ~ 22 MByte/s (175 MBit/s)**

Motivation (2)

Media and max. transfer rate versus signal type and data rate

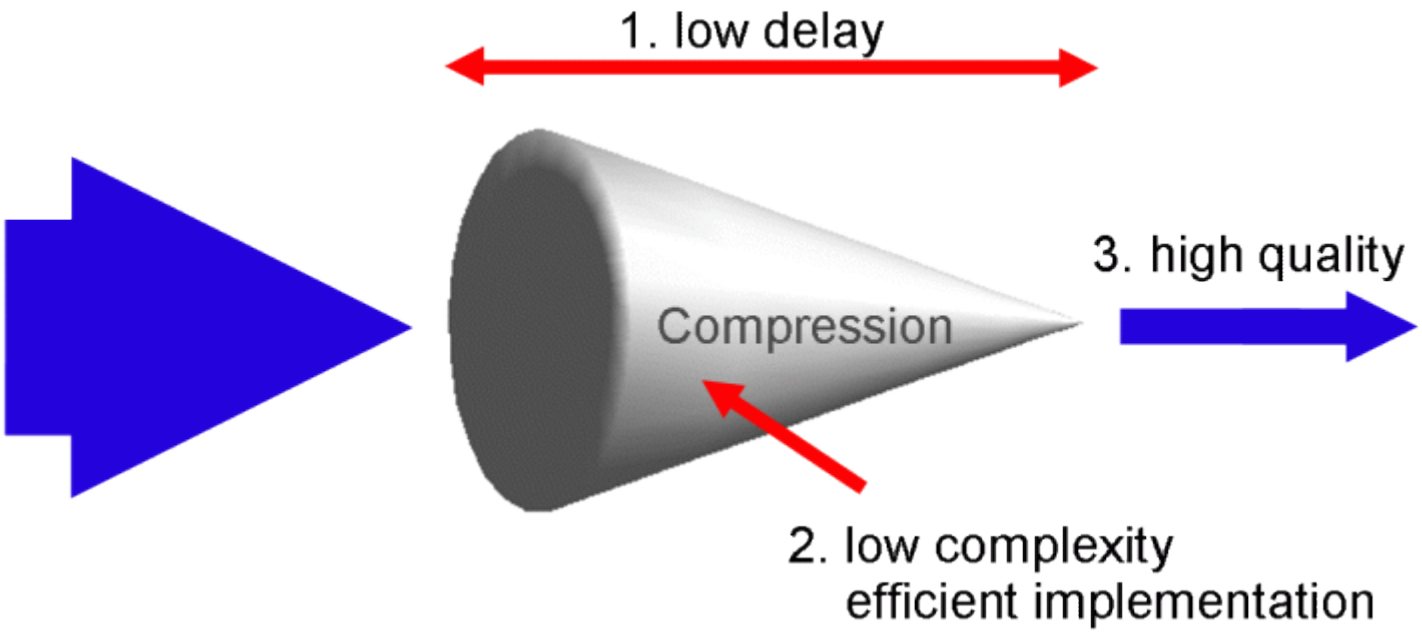
media	max. transfer rate
GSM	9.6 KBit/s
phone	28.8 KBit/s
ISDN	64 KBit/s per channel
ADSL	384 KBit/s - 8 MBit/s
CD-ROM (1x)	1,2 MBit/s
UMTS	2 MBit/s
LAN	10,100,1000 MBit/s
WLAN	11, 54 MBit/s

signal type	data rate
PAL - RGB	248,8 MBit/s
PAL - YCrCb 4:2:2	165,9 MBit/s
PAL - MPEG-1	1,18 MBit/s

Motivation (3)

- **Classification of Data**
  - redundant information
    - reconstruction without loss of data
  - irrelevant information
    - not perceived by humans
    - no need to be perceived by humans
    - take advantage of physiologic and psychological aspects of human perception
    - explicit errors possible
  - relevant information
- **Compression techniques**
  - reduce redundant and irrelevant information; keep relevant information
  - cut relevant information to achieve high compression ratio

5.2. General Requirements



- **Dependency on application type:**
  - Dialogue mode
  - Retrieval mode

Mode Dependent Requirements

- **Dialogue and retrieval mode requirements:**
  - Independence of frame size and video frame rate
  - Synchronization of audio, video, and other media
- **Dialogue mode requirements:**
  - Compression and decompression in real-time
  - End-to-end delay < 150ms
- **Retrieval mode requirements:**
  - Fast forward and backward data retrieval
  - Random access within 1/2 s

General Aspects (1)

- **Compression ratio**
  - relates the numerical representation (the data size) of the original data in comparison to the compressed data
  - CR = original data size / compressed data size
- **Lossless versus lossy compression**
  - lossless/noiseless compression techniques
    - enable exact reconstruction of data
    - exclusively reduction of redundant information
  - lossy compression techniques
    - loss of quality between original data and decompressed data
    - reduction of irrelevant or relevant information

General Aspects (2)

- **Symmetrical versus asymmetrical**
  - symmetrical compression scheme
    - decompression (decoding) requires same time as compression (encoding)
  - asymmetrical compression scheme
    - generally compression needs more time than decompression
    - asymmetry ratio denotes degree of asymmetry,
      e. g. 150:1 means it takes 150 minutes to compress one minute of video
    - can be more elaborate and more efficient for quality and speed at playback,
      e. g. Video DVD, MPEG-2
- **Realtime versus non-realtime**
  - "Realtime" means exactly what it says!
  - capture, compress and save or transmit in realtime (e.g. 25 frames per second)
  - resp. load or receive, decompress, and playback

General Aspects (3)

- **Intraframe versus interframe**
  - occuring while dealing with movie/video compression intraframe method
    - compression of single pictures (frames) independent of neighbouring frames
    - each video frame is compressed and stored as a discrete picture
    - also called "spatial compression"
  - interframe method
    - compression taking into account neighbouring frames
    - idea: backgrounds in most video scenes remain more or less stable, compress frames with respect to reference frames and transmit difference frames
    - also called "temporal compression"
- **Codec** (**Co**der and **Dec**oder)
  - actual practical solutions which process compression and decompression
  - uses one or more compression techniques
  - soft- and/or hardware solution
  - often standardized

Information theory - Entropy (1)

- **Information theory**, Shannon (1948):
  - The theory does not testify anything on the meaning of a message, it observes relationships of an information source given by its statistical structure. The information source produces information events (symbols) with determined probabilities.
  - example application: picture is source of information and pixel are information events
- Information and **self-information** (information content) of a symbol
  - Information represents news or eliminated uncertainty.
  - The information content of a symbol is represented by the uncertainty which is eliminated by the appearance of the symbol.
  - The **information content of a symbol** is the larger, the more uncertainty is eliminated by its appearance resp. the less probable its appearance is.
  - The information content (self-information) of a single symbol is defined by

$$H_i = \log_2 \frac{1}{p_i} \text{ [bit]}$$

theoretical background:
[lossless compression]
[lossy compression]
[Diskussion des Informationsbegriffs]

Information theory - Entropy (2)

- Note: the choice of the basis of *log* determines the unit of information content of a symbol
  - basis 2: ld (log. dualis) → [bit]
  - basis e: ln (log. naturalis) → [nit]
  - basis 10: lg → [dit]
  - choice of basis 2 is common

- **Information content of an information source - Entropy**
  - **Entropy** is the average information content of an information source. It measures the average uncertainty of the outcome of an information source and (in the case of statistically independent events) is defined by

$$H = \eta = \sum_i p_i \cdot \log_2 \frac{1}{p_i} \quad [\text{bit}]$$

  - Results:
    - uniform probability distribution results in maximum entropy
    - the more non-uniform the probability distribution is, the smaller the entropy is

Information theory - Entropy (3)

Examples:

- with certainty appearing symbol:
  $p_i = 1 \Rightarrow H_i = \log_2(1/p_i) = 0$
- with a low probability appearing symbol:
  $p_i = 0{,}000001 \Rightarrow H_i = \log_2(1/p_i) \approx 19{,}93$
- a picture with a uniform distribution of four gray-scale values:
  $p_i = 1/4 \Rightarrow H_i = \log_2(1/p_i) = 2$ (i=1,...,4) $H = 4 \cdot 1/4 \cdot 2 = 2$
- a picture with a strong non-uniform distribution of four gray-scale values:
  $p_i = 1/32$ for i=1,..,3 $\Rightarrow H_i = \log_2(1/p_i) = 5$ (i=1,...,3)
  $p_4 = 29/32 \Rightarrow H_4 = \log_2(1/p_i) \approx 0{,}142$
  $H \approx 3 \cdot 1/32 \cdot 5 + 29/32 \cdot 0{,}142 \approx 0{,}597$

Information theory - Entropy (4)

Entropy and Coding

- The number of bits to represent (code) a symbol *i* (one event of an information source) is called the "code word length" l_i, measured in [Bit].
- The mean value of the code word length of all symbols of an information source is called "average code word length" *l*, measured in [Bit].
- **Coding theorem - entropy coding**
  For every discrete information source with entropy *H* exists a coding scheme with average code word length *l*, so that

$$H \leq \bar{l} < H + 1$$

  holds true.
  Entropy coding schemes cannot have an average code word length smaller than *H*.
- Application: entropy coding such as huffman and arithmetic coding

Information theory - Entropy (5)

Conclusions

- Optimal coding schemes ensure that the average code word length is hardly greater or almost equal the entropy.
- There is no coding scheme with an average code word length less than the entropy of the information source except losses are admissible.
- Small information content (low uncertainty)
  denotes more non-uniform probability
  denotes small entropy
  denotes small optimal average code word length
  denotes high compression rate (and low data rate)
- Uniform or almost uniform probability distributions counteract a small entropy and a high compression ratio!

Information theory - Entropy (6)

Entropy and statistically dependent events (1)

- Definition of entropy as a measure for the amount of information generated by an information source holds true.
- Shannon's coding theorem that the best that a lossless compression scheme can do is to encode the output of a source with an average number of bits equal to the entropy of the source holds true, especially

$$H \le \overline{l} < H + 1$$

- Problem:

$$H = \eta = \sum_i p_i \cdot \log_2 \frac{1}{p_i} \quad [\text{bit}]$$

is only correct for statistically independent events, in fact for statistically dependent events our formula becomes an estimate of the entropy:

$$H < \sum_i p_i \cdot \log_2 \frac{1}{p_i} \quad [\text{bit}]$$

Information theory - Entropy (7)

Entropy and statistically dependent events (2)

- Example: Sequence: 1 2 3 2 3 4 5 4 5 6 7 8 9 8 9 10
  - assume independence:
    $p_1=p_6=p_7=p_{10}=1/16$, $p_2=p_3=p_4=p_5=p_8=p_9=2/16$
    → "H" = 3,25
  - assume structure: $x_n = x_{n-1} + r_n$
    r_n sequence: 1 1 1 -1 1 1 1 -1 1 1 1 1 1 -1 1 1
    and independence in the new sequence r_n
    $p_1=13/16$, $p_{-1}=3/16$
    → "H" ≈ 0,70
- Result: standard entropy coding techniques (such as huffman and arithmetic coding) which rely on statistically independent events are unable to satisfy

  $$H \leq \bar{l} < H + 1$$

  in the statistically dependent case (correlated data, structure in the data).
- Solution: de-correlation or pre-coding techniques

Coding

- Coding is the assignment of binary sequences to elements of an alphabet.
  - The set of binary sequences is called a code, and the individual members of the set are called codewords.
  - An alphabet is a collection of symbols called letters.
    - For example, the alphabet used in writing books consists of 26 lower- and uppercase letters, and a variety of punctuation marks. The ASCII code for the letter "a" is 1100001, the letter "A" is coded as 1000001, and the letter "," is coded as 0101100.
    - Notice that the ASCII code uses the same number of bits to represent each symbol. Such a code is called a fixed-length code. If we want to reduce the number of bits required to represent different massages, we need to use a different number of bits to represent different symbols. If we use fewer bits to represent symbols that occur more often, on the average we would use fewer bits per symbol.
    - The average number of bits per symbol is often called the rate of the code.

Coding: The Morse Code

- The idea of using fewer bits to represent symbols that occur more often is used in Morse code:

  the codewords for letters that occur more frequently are shorter than for letters that occur less frequently. For example, the codeword for "E" is •, while the codeword for "Z" is – – • •.

Letter	probability German	probability English	Morse Code
e	16,65%	12,41%	•
n	10,36%	6,41%	– •
i	8,14%	6,46%	• •
t	5,43%	8,90%	–
a	5,15%	8,09%	• –
o	2,25%	8,13%	– – –
x	0,03%	0,20%	– • • –
y	0,03%	2,14%	– • – –

Uniquely Decodeable Codes

- The average length of the code is not the only important point in designing a "good" code.
- Example: Suppose our source alphabet consists of four letters a_1, a_2, a_3 and a_4, as follows:

Letters	Probability	Code 1	Code 2	Code 3	Code 4
a_1	0,5	0	0	0	0
a_2	0,25	0	1	10	01
a_3	0,125	1	00	110	011
a_4	0,125	10	11	111	0111
Average Length		1,125	1,25	1,75	1,875

- The entropy for this source is 1.75 bits/symbol and the average length l for each code is given by: $l = \sum_{i=1}^{4} P(a_i) n(a_i)$

  where $n(a_i)$ is the number of bits in the codeword for letter a_i and the average length is given bits/symbol.

Coding

- Code 1

Based on the average length, Code 1 appears to be the best code. However, to be useful, a code should have the ability to transfer information in an unambiguous manner. This is obviously not the case with Code 1. Both a_1 and a_2 have been assigned the codeword 0. When a 0 is received, there is no way to know whether an a1 was transmitted or an a_2. We would like each symbol to be assigned a unique codeword.

- Code 2

Suppose we want to encode the sequence $a_2\ a_1\ a_1$. Using Code 2, we would encode this with the binary string 100. However, when the string 100 is received at the decoder, there are several ways in which the decoder can decode this string. The string 100 can be decoded as $a_2\ a_1\ a_1$ or as $a_2\ a_3$.

This means that once a sequence is encoded with Code 2, the original sequence cannot be recovered with certainty.

In general, this is not a desirable property for a code. We would like unique decodability from the code; that is, any given sequence of codewords can be decoded in one, and only one, way.

Coding

- Code 3

We have already seen that Code 1 and Code 2 are not uniquely decodable. How about Code 3? Notice that the first three codewords all end in a 0. In fact, a 0 always denotes the termination of a codeword. The final codeword contains no 0´s and is 3 bits long. Because all other codewords have fewer than three 1s and terminate in a 0, the only way we can get three 1´s in a row is as a code for a_4. The decoding rule is simple. Accumulate bits until you get a 0 or until you have three 1´s. There is no ambiguity in this rule, and it is reasonably easy to see that this code is uniquely decodable.

- Code 4

Each codeword starts with a 0, and the only time we see a 0 is in the beginning of a codeword. Therefore, the decoding rule is accumulate bits until you see a 0. The bit before the 0 is the last bit of the previous codeword.

There is a slight difference between Code 3 and Code 4. In the case of Code 3, the decoder knows the moment a code is complete. In Code 4, we have to wait till the beginning of the next codeword before we know that the current codeword is complete. Because of this property, Code 3 is called an instantaneous code. Although Code 4 is not an instantaneous code, it is almost that.

Uniquely Decodeable Codes: Examples

While this property of instantaneous or decoding is a nice property to have, it is not a requirement for unique decodability.

- Code 5

Consider the following code:

Code 5	
Letter	Codeword
a_1	0
a_2	01
a_3	11

Let's decode the string:

0111111111111111

In this string, the first codeword is either 0 corresponding to a_1 or 01 corresponding to a_2. We cannot tell which one until we have decoded the whole string. Starting with the assumption that the first codeword corresponds to a_1, the next eight pairs of bits are decoded as a_3.

However, after decoding eight a_3´s, we are left with a single (dangling) 1 that does not correspond to any codeword. On the other hand, if we assume the first codeword corresponds to a_2, we can decode the next 16 bits as a sequence of eight a_3´s, and we do not have any bits left over. The string can be uniquely decoded. In fact, Code 5, while it is certainly not instantaneous, is uniquely decodable.

Coding

We found that we had an incorrect decoding because we were left with a binary string (1) that was not a codeword. If this had not happened, we would have had two valid decodings.

- Code 6

For example, consider the following code:

Code 6	
Letter	Codeword
a_1	0
a_2	01
a_3	10

Let's encode the sequence a_1 followed by eight a_3´s using this code. The coded sequence is:

0101010101010101

The first bit is the codeword for a_1. However, we can also decode it as the first bit of the codeword for a_2.

If we use this (incorrect) decoding, we decode the next seven pairs of bits as the codewords for a_2. After decoding seven a_2s, we are left with a single 0 that was decode as a_1. This the incorrect decoding is also a valid decoding, and this code is not uniquely decodable.

Coding

- Definition:
  - Suppose two binary codewords a and b, where a is k bits long, b is n bits long, and k < n.
    - If the first k bits of b are identical to a, then a is called a prefix of b.
    - The last n - k bits of b are called the dangling suffix.
    - For example, if a = 010 and b = 01011, then a is a prefix of b and the dangling suffix is 11.

A Test for Unique Decodability (II)

- Test:
  - Construct a list of all codewords.
    - Examine all pairs of codewords to see if any codeword is a prefix of another codeword.
    - Whenever you find such a pair, add the dangling suffix to the list unless you have added the same dangling suffix to the list in a previous iteration.
    - Now repeat the procedure using this larger list. Continue in this fashion until one of the following two things happens:

  1. You get a dangling suffix that is a codeword.
  2. There are no more unique dangling suffixes.

- If you get the first outcome, the code is not uniquely decodable. However, if you get the second outcome, the code is uniquely decodable.

A Test for Unique Decodability (III)

- Code 5

First list the codewords:

{0,01,11}

The codeword 0 is a prefix for the codeword 01. The dangling suffix is 1. There are no other pairs for which one element of the pair is the prefix of the other. Let us augment the codeword list with the dangling suffix:

{0,01,11,1}

Comparing the elements of this list, we find 0 is a prefix of 01 with a dangling suffix of 1. But we have already included 1 in our list. Also, 1 is a prefix of 11. This gives us a dangling suffix of 1, which is already in the list. There are no other pairs that would generate a dangling suffix, so we cannot augment the list any further. Therefore, Code 5 is uniquely decodable.

A Test for Unique Decodability (IV)

- Code 6

First list the codewords

{0,01,10}

The codeword 0 is a prefix for the codeword 01. The dangling suffix is 1. There are no other pairs for which one element of the pair is the prefix of the other.

Augmenting the codeword list with 1, we obtain the list

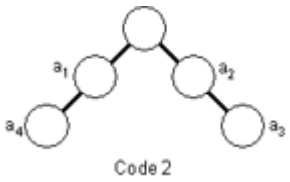
{0,01,10,1}

In this list, 1 is a prefix for 10. The dangling suffix for this pair is 0, which is the codeword for a1.

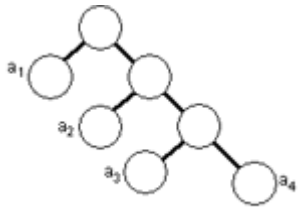
Therefore, Code 6 is not unique decodable.

Prefix Codes

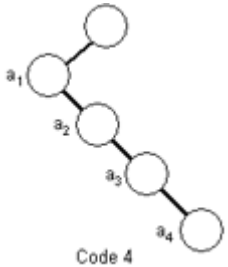
- A prefix code is a code:
  - in which no codeword is a prefix of the other. In this case, the set of dangling suffixes is the null set, and code in which no codeword is a prefix to another codeword is called a prefix code.
  - How to check for a prefix code:
    - A simple way to check it is to draw the rooted binary tree of the code. The tree starts from the root node and has two possible branches at each node, one corresponds to a 1 and the other to a 0.
    - Convention: we start with the root node at the top, the left branch corresponds to a 0 and the right branch corresponds to a 1. Using this convention, we can draw the binary tree for Code 2, Code 3, and Code 4



Code 2



Code 3



Code 4

Categories and Techniques (1)

entropy coding	repetitive sequence suppression	zero supression
		run-length coding
	statistical coding	Morse coding
		Huffman coding
		arithmetic coding
		pattern substitution / dictionary techniques (e.g. LZ77, LZ78, LZW)
source coding	predictive coding / differential coding	DPCM
		DM
	layered coding	subband coding
		subsampling
	transform coding	color space
		FFT
		DCT
		wavelet decomposition
	quantization	scalar quantization
		vector quantization (fractal etc.)

In practice, entropy coding and source coding are used in conjunction: "hybrid coding".

Categories and Techniques (2)

- **Entropy Coding Techniques**
  assigns codes to symbols so as to match code lengths with the probabilities of the symbols
  - Run Length Encoding (RLE) codes sequences in which the same data value occurs in many consecutive data elements as a single data value and the number of occurences. Zero Supression is a special case of RLE.
  - Huffman and Arithmetic Coding code different symbols or constellations of symbols according to their relative frequency in the data.
    They assume statistically independent events from the source.
    Both Huffman and Arithmetic coding are optimal entropy encoding techniques in terms of satisfying the coding theorem.
  - Dictionary techniques incorporate the structure in the data and assume statistically dependent events from the data source

Categories and Techniques (3)

- **Source Coding Techniques**
  take into account specific characteristics of the data source
  - Predictive Coding/ Differential Coding:
    Makes use of the (past) history of the data in order to increase the amount of compression.
    Sources such as speech and images have a great deal of correlation from sample to sample. This fact is used to predict each sample based on its past and only encode and transmit the differences between the prediction and the sample value.
  - Layered Coding:
    Layered Coding divides data into different levels of hierarchy and treats these data differently
  - Subband coding relies on separation of the source data into different bands of frequencies using digital filters. Subsampling reduces the number of samples in a sequence by simply omitting samples, by averaging neighboring samples or by other filter operations.
    In image processing, "chroma subsampling" is the use of lower resolution for the colour information in an image than for the brightness information.

Categories and Techniques (4)

- **Source Coding Techniques (cont.)**
  - Transform Coding:
    Transform data into a different representation in order to separate data into parts of different relevance.
    Pure Transform Coding schemes are lossless techniques.
  - Quantization:
    Reducing the number of bits to represent a symbol/sample by quantization of real to integer values (e.g. plus division by factors) or e.g. simply by discarding least significant bits.

    Fractal coding uses mathematical functions to represent data.
- **Hybrid Coding**
  Using several different coding techniques together. In practice, this is the most frequent approach.