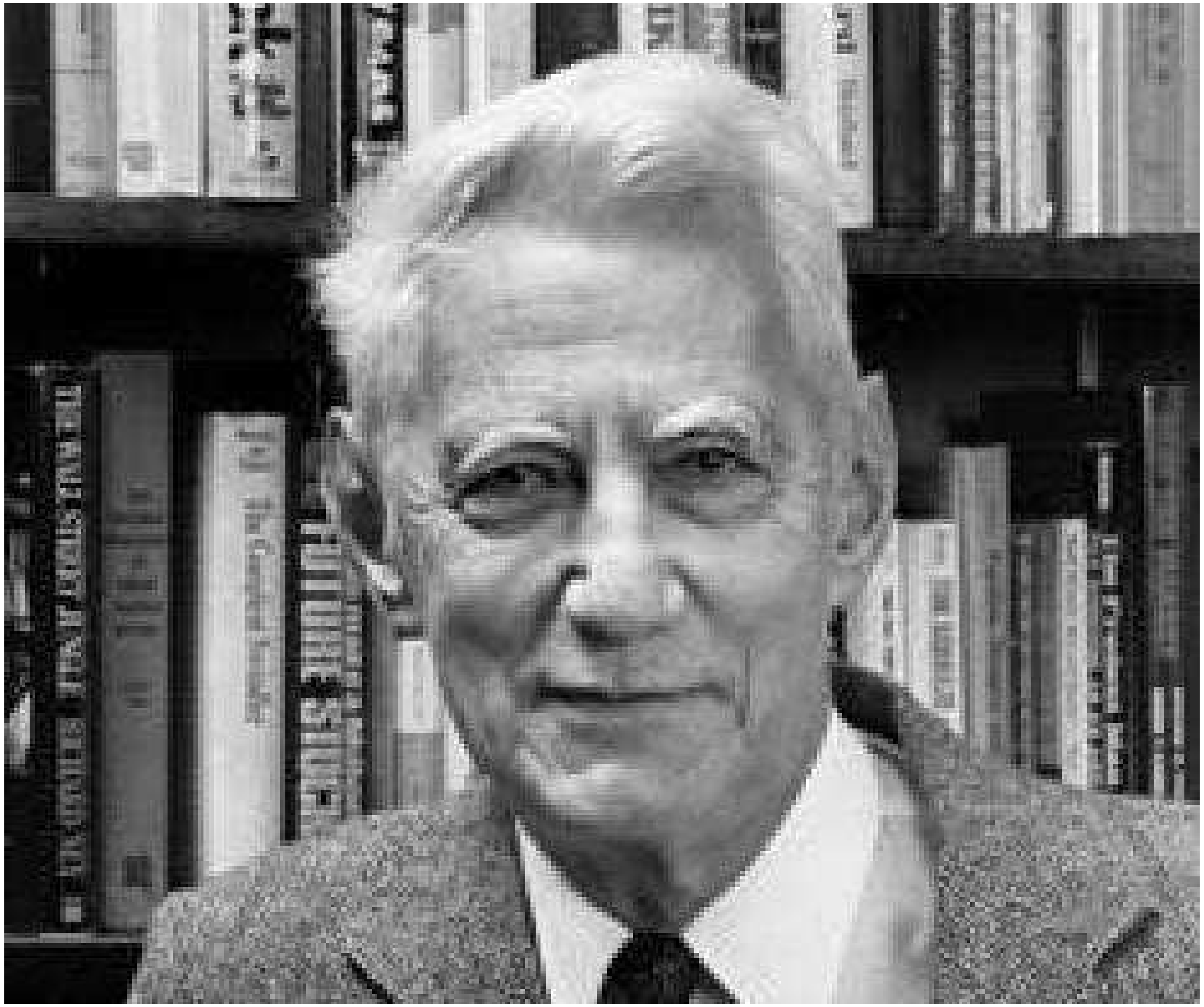
Data compression
&
Information theory

Gjerrit Meinsma

Claude Elwood Shannon (1916-2001)



Outline

1. Data compression
2. Universal compression algorithm
3. Mathematical model for ‘disorder’ (information & entropy)
4. Connection entropy and data compression
5. ‘Written languages have high redundancy’
   (70% can be thrown away)

Data compression

For this LaTeX file:

$$\frac{\#\ \text{bytes after compression (gzip)}}{\#\ \text{bytes before compression}} = 0.295$$

Some thoughts:

1. Can this be beaten?
2. Is there a 'fundamental compression ratio'?

Two extremes

Theorem 1. *De optimal compression ratio for this file* < 0.001.

Two extremes

Theorem 1. *De optimal compression ratio for this file* < 0.001*.*

....because my own compression algorithm `gjerritzip` assigns a number to each file and this LaTeX-file happens to get number 1.

Two extremes

Theorem 1. *De optimal compression ratio for this file* < 0.001.

....because my own compression algorithm `gjerritzip` assigns a number to each file and this LaTeX-file happens to get number 1.

`gjerritzip` probably doesn't do too well on other files.

We want a universal compression algorithm
one such that for every file

$$\frac{\#\text{ bytes after compression}}{\#\text{ bytes before compression}} \leq \alpha$$

witjh $\alpha > 0$ as small as possible.

We want a universal compression algorithm
one such that for every file

$$\frac{\#\ \text{bytes after compression}}{\#\ \text{bytes before compression}} \leq \alpha = 1$$

witjh $\alpha > 0$ as small as possible.

Unfortunately:

Theorem 2.
There is no lossless compression that strictly reduces every file.

Proof.

Consider the two one-bit files ‘0’ en ‘1’.....

Proof.

Consider the two one-bit files '0' en '1'.....

More convincing:

There are 2^N files of N bits.

There are

$$2^{N-1} + 2^{N-2} + \cdots 2^0 = 2^N - 1$$

files of less than N bits. ■

Funny: patents have been granted to universal compression algorithms, which we know don't exist.

Exploiting structure

Why do `gzip`, `winzip` work in practice?

Exploiting structure

Why do `gzip`, `winzip` work in practice?

Because computer files have a structure.

`gzip` works for many, many files, but not for every file.

Exploiting structure

Why do `gzip`, `winzip` work in practice?

Because computer files have a structure.

`gzip` works for many, many files, but not for every file.

...it's time to define 'structure' mathematically

...the notion of information & entropy

Towards a definition of information & entropy

Compare

‘I will eat something today’

with

‘Balkenende has a lover’

The second one is more surprising, supplies more information.

- Information is a measure of surprise
- (Entropy is a measure of disorder)

What do we want ‘information’ to be?

Example 3. A draw a card. Then I tell you:

Example 3. A draw a card. Then I tell you:

- ‘It is a spade’ [I supply information]

Example 3. A draw a card. Then I tell you:

- ‘It is a spade’ [I supply information]
- ‘It is an ace’ [I supply information once again]

Example 3. A draw a card. Then I tell you:

- 'It is a spade' [I supply information]
- 'It is an ace' [I supply information once again]

It seems reasonable to demand that

$$\mathrm{I}(\text{ace of spade}) = \mathrm{I}(\text{spade}) + \mathrm{I}(\text{ace})$$

That is to say:

$$\mathrm{I}(A \cap B) = \mathrm{I}(A) + \mathrm{I}(B) \text{ if } A \text{ and } B \text{ are indepedent}$$

Axiom 4 (Version 1).

1. $\mathrm{I}(A \cap B) = \mathrm{I}(A) + \mathrm{I}(B)$ if A and B independent
2. $\mathrm{I}(A) \geq 0$
3. $\mathrm{I}(A) = \mathrm{I}(B)$ if $P(A) = P(B)$

Because of the last axiom, information is a function of probability:

Axiom 5 (Version 2). For all $p, q \in (0, 1)$:

1. $\mathrm{I}(pq) = \mathrm{I}(p) + \mathrm{I}(q)$
2. $\mathrm{I}(p) \geq 0$
3. and we add anoter: $\mathrm{I}(p)$ is continuous in p.

Theorem 6.
Then I *is unique:* $\mathrm{I}(p) = -k\log(p)$ *modulo scaling* $k > 0$*.*

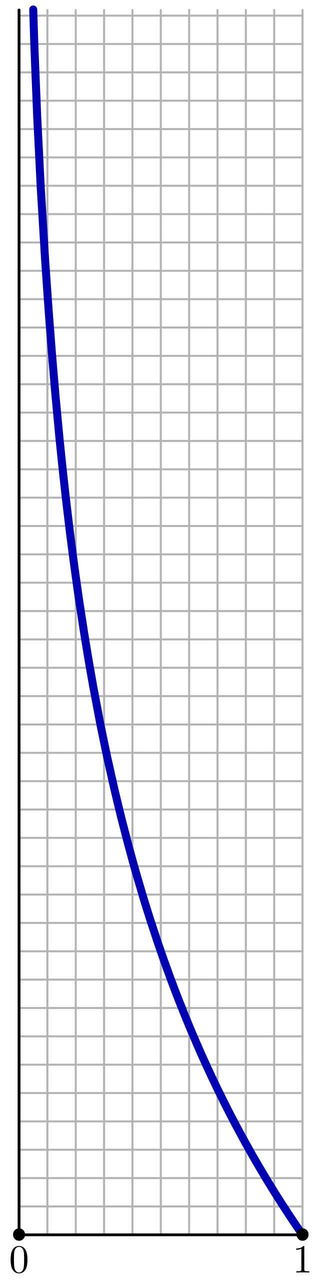
Other scaling factor k means other unit (irrelevant).

From now on:

$$\boxed{\mathrm{I}(p) = -\log_2(p)}$$

Example 7 (Special cases).

- $\mathrm{I}(0) = +\infty$, makes sense
- $\mathrm{I}(1) = 0$, makes sense
- $\mathrm{I}(2^{-k}) = k$, well...



Entropy = expectation of information

Definition 8. Entropy $H := \mathbb{E}(I)$

Entropy = expectation of information

Definition 8. Entropy $H := \mathbb{E}(I)$

Example 9 (Connection entropy and structure).
Consider the file

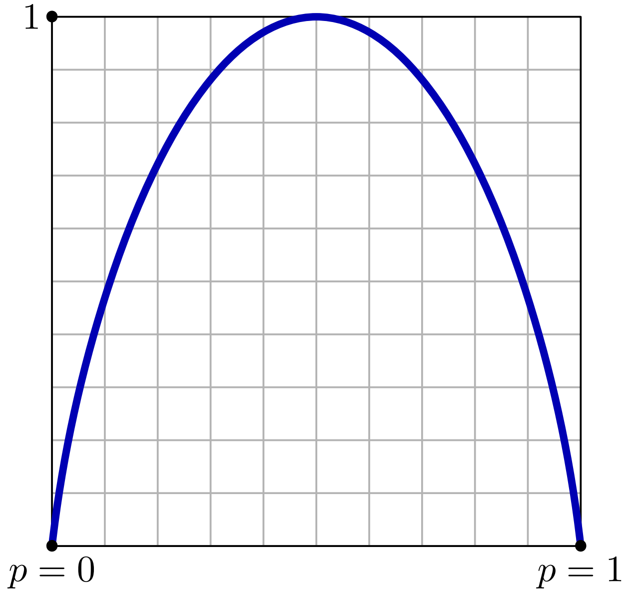
aaaaabaabbaaaaababaaaaabaaaaaabaabaaab....

and assume

- $P(\text{a}) = p$
- $P(\text{b}) = 1 - p$

then entropy per symbol is

$$H = \mathbb{E}(I) = p\,\mathrm{I}(p) + (1-p)\,\mathrm{I}(1-p)$$



Makes sense:

$p \approx 1:$	aaaaabaaaaaabaaaa	little surprise
$p \approx 0:$	bbbbbbbabbbbbbab	little surprise
$p = 1/2:$	abaabbabaabbaaba	maximal disorder (symmetry)

Example 10 (The abcd-file).
Consider file with symbols ‘a’, ‘b’, ‘c’ en ‘d’:

bdaccdaadaabbcaaabbaaabaaacbdaab.....

and suppose that

$$
\begin{aligned}
P(\mathsf{a}) &= 1/2 \\
P(\mathsf{b}) &= 1/4 \\
P(\mathsf{c}) &= 1/8 \\
P(\mathsf{d}) &= 1/8
\end{aligned}
$$

Its entropy (per symbol) is

$$
\begin{aligned}
H &= \sum_i p_i \,\mathrm{I}(p_i) \\
&= \frac{1}{2}\,\mathrm{I}(2^{-1}) + \frac{1}{4}\,\mathrm{I}(2^{-2}) + 2\frac{1}{8}\,\mathrm{I}(2^{-3}) \\
&= \frac{1}{2}(1) + \frac{1}{4}(2) + 2\frac{1}{8}(3) \\
&= \frac{1}{2} + \frac{1}{2} + \frac{3}{4} \\
&= 1.75
\end{aligned}
$$

Back to compression

Example 11 (The abcd-file). Consider again

bdaccdaadaabbcaaabbaaabaaacbdaab.....

with again

$$\begin{aligned} P(\mathsf{a}) &= 1/2 \\ P(\mathsf{b}) &= 1/4 \\ P(\mathsf{c}) &= 1/8 \\ P(\mathsf{d}) &= 1/8 \end{aligned}$$

This we want to code (binary)

This is an obvious coding:

$$\begin{array}{lcl} \text{a} & \leftrightarrow & 00 \\ \text{b} & \leftrightarrow & 01 \\ \text{c} & \leftrightarrow & 10 \\ \text{d} & \leftrightarrow & 11 \end{array}$$

code word

For example:

$$\text{abba} \quad \leftrightarrow \quad 00010100$$

This coding is decodable

As all code words $\in \{00, 01, 10, 11\}$ consist of two bits:

$$\text{(average) \# bits per symbol} = 2$$

As all code words $\in \{00, 01, 10, 11\}$ consist of two bits:

$$\text{(average) \# bits per symbol} = 2$$

Can this be done more efficiently?

Yes:

$$\begin{array}{lllr} \text{a} & \leftrightarrow & 0 & \text{with probability } 1/2 \\ \text{b} & \leftrightarrow & 10 & \text{with probability } 1/4 \\ \text{c} & \leftrightarrow & 110 & \text{with probability } 1/8 \\ \text{d} & \leftrightarrow & 111 & \text{with probability } 1/8 \end{array}$$

Symbols that appear more often have shorter code words

Yes:

a	$\leftrightarrow$	0	with probability $1/2$
b	$\leftrightarrow$	10	with probability $1/4$
c	$\leftrightarrow$	110	with probability $1/8$
d	$\leftrightarrow$	111	with probability $1/8$

Symbols that appear more often have shorter code words

$$\begin{aligned}\text{average \# bits per symbol} &= \frac{1}{2}(1) + \frac{1}{4}(2) + 2\frac{1}{8}(3) \\ &= 1.75\end{aligned}$$

Can this be done still more efficiently?

Can this be done still more efficiently?

Shannon (1948): ‘No, cause entropy of the source is 1.75’

Theorem 12.

$$H \leq \inf_{\textit{codings}} \mathbb{E}(L) \leq H + 1.$$

 # bits per symbol

Example 13 (terrible coding). Suppose our file is

aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

with $P(\mathsf{a}) = 1$. This file has entropy (per symbol)

$$H = 0.$$

and for the 'optimal' coding

$$\mathsf{a} \quad \leftrightarrow \quad 0$$

we have

$$\mathbb{E}(L) = 1 = H + 1$$

Example 13 (terrible coding). Suppose our file is

aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

with $P(\mathsf{a}) = 1$. This file has entropy (per symbol)

$$H = 0.$$

and for the 'optimal' coding

$$\mathsf{a} \quad \leftrightarrow \quad 0$$

we have

$$\mathbb{E}(L) = 1 = H + 1$$

Message: Coding per symbol is not a good idea.

Coding sequence of symbols

Not allowed was the coding

aaaaaaaaaaaaaaaaaaaaaaaaa $\leftrightarrow$ 25 a's $\leftrightarrow$ ....

Pity, but not too bad:

We may consider

absdfuhquwyhgvsdfvsefqwsddfhwdqqwsd

as a sequence of 'letters' (symbols)
but also as a sequence of N-letters (symbols)

absdfuhquwyhgvsdfvsefqwsddfhwdqqwsd

This requires coding of many 'symbols'

$$\begin{array}{ccc} \text{absdf} & \leftrightarrow & 0110101000 \\ \vdots & \leftrightarrow & \vdots \end{array}$$

Entropy per N-letter symbol is

$$H_N = NH$$

So Shannon says:

bits per symbol (N-letters)

$$NH \;\leq\; \inf \mathbb{E}(L) \;\leq\; NH + 1$$

that is

$$H \;\leq\; \inf \mathbb{E}(L/N) \;\leq\; H + 1/N$$

Beautiful:

Theorem 14.

$$\inf_{\textit{codings}} \; \mathbb{E}(\#\ \textit{bits per letter}) \;=\; H$$

Entropy of languages—redundancy

Written languages have high redundancy:

> Aoccdrnig to rscheearch at an Elingsh uinervtisy, it deosn't mttaer in waht oredr the ltteers in a wrod are, the olny iprmoetnt tihng is taht the frist and lsat ltteer is at the rghit pclae.

Languages are highly structured.

It doesn't seem to have high entropy.

How to determine the entropy of 'English'

Silly approach:

$$P(\mathsf{a}) = P(\mathsf{b}) = \cdots = P(\mathsf{z}) = P(\ ) = \frac{1}{27} \implies H \approx 4.8$$

First-order approximation:

$$\begin{aligned} P(\mathsf{a}) &= 0.0575 \\ P(\mathsf{b}) &= 0.0128 \\ &\vdots \\ P(\mathsf{e}) &= 0.0913 \\ &\vdots \\ P(\mathsf{z}) &= 0.007 \end{aligned}$$

then

$$H \approx 4.1$$

Higher-order approximation:

$$
\begin{aligned}
P(\text{be} \,|\, \text{to be or not to}) &= ?? \\
\vdots \quad & \quad \vdots
\end{aligned}
$$

Then

$$H = ?$$

It is believed that

$$0.6 < H_{\mathsf{English}} < 1.3$$

that is: about 1 bit per letter is enough!

Computers represent letters in ASCII (8-bits)
so this compression ratio should be possible (in theory):

$$\frac{H_{\mathsf{English}}}{8} < \frac{1.3}{8} \approx \frac{1}{6}$$

In other words:

Perfect zippers compress ‘English’ with a factor of 6.

Proofs

Lemma 15. *For every decodable coding*

$$\sum_i 2^{-l_i} \leq 1$$

with l_i *the length of code word number* i.

Proof.

$$\begin{array}{ccc} \overbrace{\textsf{aaaaaaaaa}}^{N} & \leftrightarrow & \overbrace{1010001011}^{\sum_{k=1}^{N} l_{i_k}} \\ \textsf{baaaaaaaa} & \leftrightarrow & 0110110001111 \\ \vdots & \vdots & \vdots \\ \textsf{zzzzzzzzz} & \leftrightarrow & 011011010001111 \end{array}$$

Decodable implies

$$\#\ \text{codewords of length } l := \sum_{k=1}^{N} l_{i_k} \leq 2^l$$

$$S_N := (\sum_i 2^{-l_i})^N = \sum_{i_1, i_2, \ldots, i_N} 2^{-(l_{i_1} + \cdots + l_{i_N})}$$

then

$$S_N = \sum_l 2^{-l} A_l, \qquad A_l := \# \text{ of length } l$$

Then $A_l \leq 2^l$ because uniquely decodable. So

$$S_N = \sum_l 2^{-l} A_l \leq \sum_l 2^{-l} 2^l = N l_{\mathsf{max}}$$

Therefore

$$(\sum_i 2^{-l_i})^N \leq N l_{\mathsf{max}}$$

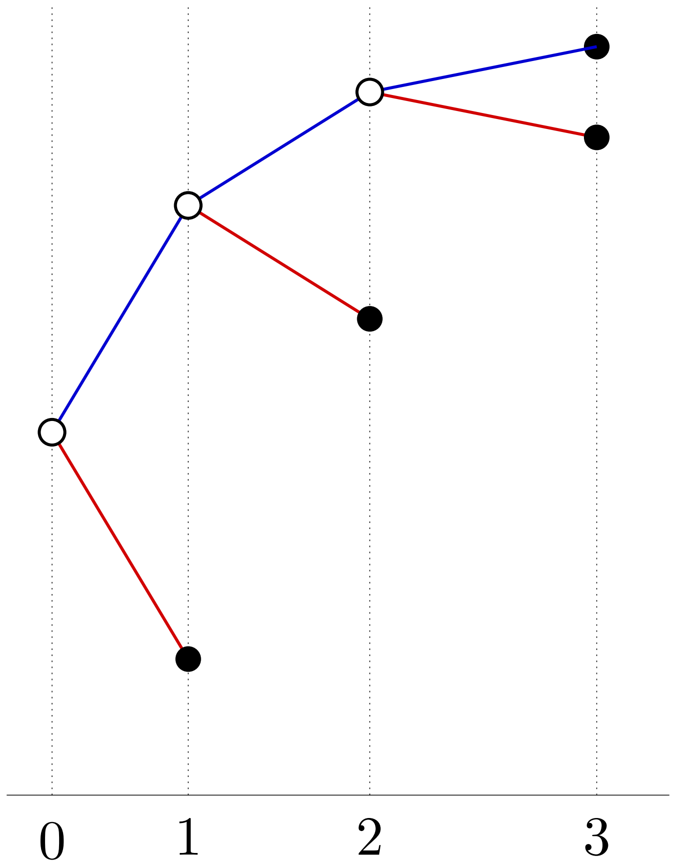
left-hand side is exponential in N, right-hand side linear.
Hence base of exponent ≤ 1. ■

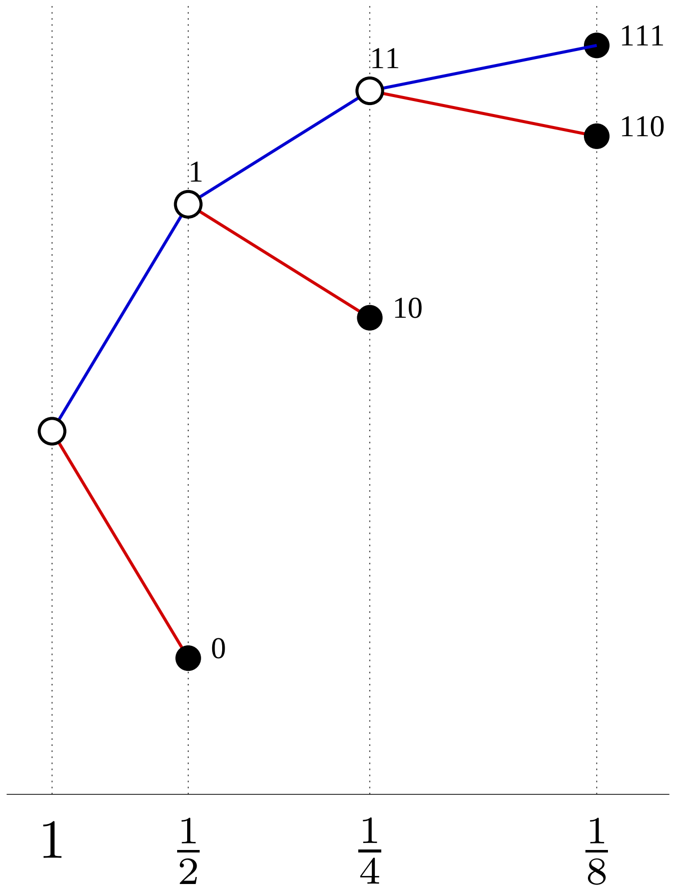
Lemma 16. *If* l_i *such that*

$$\sum_i 2^{-l_i} \leq 1$$

then there is a (direct) decodable coding.

Proof. Construct a tree with those lengths
(suppose lengths are $\{1, 2, 3, 3\}$)





This one is decodable. ■

Proof that $H \leq \mathbb{E}(L)$.

$$\begin{aligned}
\mathbb{E}(L) - H &= \sum_i p_i l_i + \sum_i p_i \log_2(p_i) \\
&= \sum_i p_i \log_2(2^{l_i} p_i) \\
&= \sum_i p_i \frac{\ln(2^{l_i} p_i)}{\ln(2)} \\
&= \frac{1}{\ln(2)} \sum_i p_i \ln(2^{l_i} p_i) \\
&\geq \frac{1}{\ln(2)} \sum_i p_i (1 - 2^{-l_i}/p_i) \\
&= \frac{1}{\ln(2)} \sum_i p_i - 2^{-l_i} \qquad \geq 0.
\end{aligned}$$

$\ln(x) \geq 1 - \frac{1}{x}$

Proof of

$$\inf \mathbb{E}(L) \leq H + 1$$

is constructive: take

$$l_i = \lceil -\log_2(p_i) \rceil$$

then

$$\sum_i 2^{-l_i} \leq \sum p_i = 1$$

so a decodable coding exists.

We have

$$\mathbb{E}(L) = \sum p_i l_i < \sum p_i(-\log_2(p_i) + 1) = H + 1.$$

■