

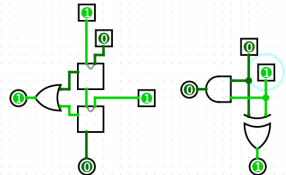
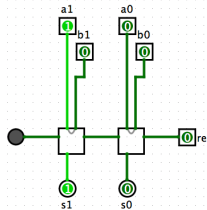
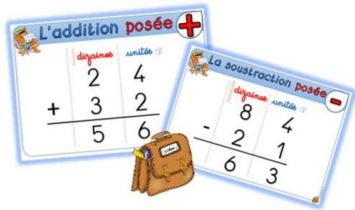
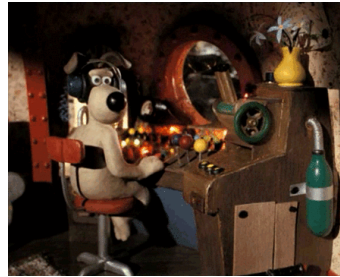
Concepts Informatiques I

2025–2026

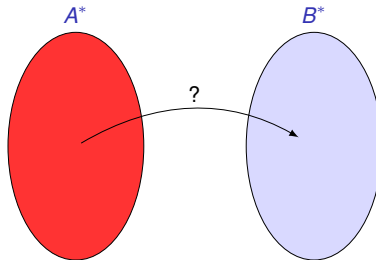
Matthieu Picantin



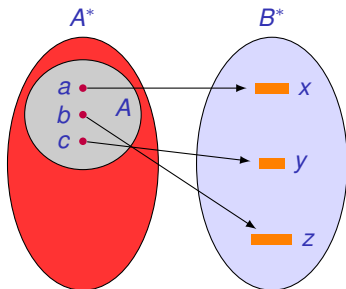
- ◆ systèmes de numération & arithmétique associée
- ◆ représentation des nombres & arithmétique machine
- ◆ codes, codages, entropie, compression
- ◆ contrôle d'erreur (détection, correction)
- ◆ crypto (confidentialité, authenticité, intégrité)
- ◆ logique et calcul propositionnel
- ◆ circuits combinatoires



Comment passer d'une représentation par des mots sur un alphabet donné A en une représentation par des mots sur un alphabet donné B ?



- ♦ représenter des objets dans un système numérique (codage)
- ♦ économiser de l'espace / de la bande passante (compression)
- ♦ résister aux altérations, pertes ou mutations (contrôle d'erreur)
- ♦ rendre illisibles aux non-initiés des données (cryptographie)



On définit un codage des lettres de A en des mots de B^* à l'aide d'une fonction $\tau : A \rightarrow B^*$.

Le **codage** par τ d'un mot $m = m_0 m_1 \cdots m_{\ell-1}$ de A^* consiste alors à coder chaque lettre et à concaténer les codages dans l'ordre :

$$\tau(m) = \tau(m_0)\tau(m_1) \cdots \tau(m_{\ell-1}).$$

(τ est alors un **morphisme** de A^* dans B^*)

On doit pouvoir retrouver le mot original : le morphisme τ doit être **injectif** !

Autrement dit, l'ensemble $\mathcal{C} = \tau(A) = \{ \text{orange}, \text{orange}, \text{orange} \}$ doit être un **code** : tout mot de \mathcal{C}^* doit se décomposer d'une seule façon sur \mathcal{C} .

Exemples avec $A = \{a, b, c\}$ et $B = \{0, 1\}$

$$\begin{cases} \tau_1(a) = 00 \\ \tau_1(b) = 11 \\ \tau_1(c) = 111110 \end{cases} \quad \begin{array}{l} \tau_1 \text{ définit bien} \\ \text{un codage injectif} \end{array}$$

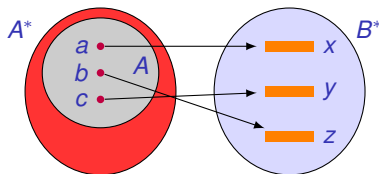
$\Leftrightarrow \mathcal{C}_1 = \{00, 11, 111110\}$ est un **code**

$$\begin{cases} \tau_2(a) = 0 \\ \tau_2(b) = 01 \\ \tau_2(c) = 10 \end{cases} \quad \begin{array}{l} \tau_2 \text{ définit} \\ \text{un codage} \\ \text{non injectif} \end{array}$$

quid de 010 : est-ce 0 10 ou 01 0 ?

- Si les images des lettres de A par τ sont toutes d'une même longueur k , le code $\tau(A)$ est dit de **longueur fixe**

$$\exists k : \tau(A) \subseteq B^k$$



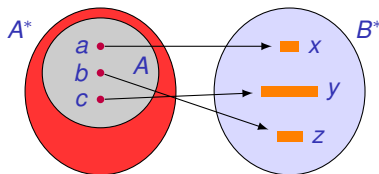
- Il faut/suffit d'avoir $k \geq \log_{|B|}(|A|)$ pour coder A sur B^* (puis coder A^* sur B^*)

- $A = \{a, b, c\}$ et $B = \{0, 1\}$ donnent $k \geq 2$, par exemple $\begin{cases} \tau_3(a) = 00 \\ \tau_3(b) = 01 \\ \tau_3(c) = 10 \end{cases}$
- $A = \{a, b, \dots, z\}$ et $B = \{0, 1\}$ donnent $k \geq 5$, par exemple $\begin{cases} \tau_4(a) = 00000 \\ \vdots \\ \tau_4(z) = 11001 \end{cases}$
- $A = \{A, B, \dots, Z, a, b, \dots, z, 0, 1, \dots, 9\}$ et $B = \{0, 1\}$ donnent $k \geq 6$, etc.

- Le décodage est facile à partir du découpage du mot image en blocs de k lettres

- $0100001000 = 01\ 00\ 00\ 10\ 00$ est le codage par τ_3 de *baaca*
- $0100001000 = 01000\ 01000$ est le codage par τ_4 de *hh*

- ♦ Si les images des lettres de A par τ ne sont pas toutes de même longueur, le code $\tau(A)$ est de **longueur variable**



- ▶ utiles si la fréquence des symboles de A n'est pas uniforme
- ▶ en général plus difficiles à construire et/ou à décoder
- ♦ Parmi eux, les codes préfixes sont les plus faciles à construire et à décoder

Un langage **préfixe** est un langage dans lequel aucun mot n'est le préfixe d'un autre mot

Tout langage préfixe est un code !

$\{0, 10, 110, 1110\}$ est un exemple de code préfixe

$\{0, 01, 011, 0111\}$ n'est pas préfixe (mais c'est un code suffixe !)

Exemple de code de longueur variable

International Morse Code

1. The length of a dot is one unit.
2. A dash is three units.
3. The space between parts of the same letter is one unit.
4. The space between letters is three units.
5. The space between words is seven units.

A ••—
 B —•••
 C —•—•
 D —•••
 E •—
 F ••—•
 G —•—•
 H ••••
 I ••—
 J •—•—
 K —•—•
 L —•••
 M —•—
 N —••
 O —•—•
 P —•—••
 Q —•—•—
 R —•••
 S •••—
 T —•—

U ••—
 V •••—
 W •—•—
 X —••—
 Y —••—
 Z —•—••

1 •—•—•—
 2 ••—•—•—
 3 •••—•—
 4 ••••—
 5 •••••
 6 —•••••
 7 —•—•••
 8 —•—••••
 9 —•—•—••
 0 —•—•—•—

Exemple de code de longueur fixe

ASCII étendu

ISO/CEI 8859-15																
	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	non utilisé															
1x																
2x		!	"	#	\$	%	&	'	()	*	+	,	-	.	/
3x	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4x	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5x	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_
6x	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7x	p	q	r	s	t	u	v	w	x	y	z	{		}	~	
8x	non utilisé															
9x																
Ax		ı	ç	£	€	¥	Š	š	©	ª	«	¬		®	ˆ	
Bx	°	±	²	³	Ž	μ	¶	·	ž	¹	º	»	ƒ	œ	ÿ	ı
Cx	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
Dx	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
Ex	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
Fx	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

Un **code compresseur** permet d'obtenir une représentation plus compacte

- ◆ en vue de transmission (économie de bande passante)
- ◆ en vue de stockage (économie d'espace)

Compression **conservative** ou **sans perte** : le message originel peut être reconstruit à l'identique en inversant la fonction

- ◆ ce type de compression est recherché avec du texte par exemple

Compression **non conservative** ou **avec perte** : le message originel n'est pas reconstruit à l'identique, mais un message similaire est obtenu à l'inversion

- ◆ souvent le cas des images et des sons : il n'est pas nécessaire que des détails quasi-invisibles-sensibles soient conservés ou transmis
- ◆ cette compression permet d'obtenir de très bons taux de compression

Quotient de compression

♦ $q = \text{volume initial} / \text{volume final}$

plus une compression sera forte,

plus le quotient de compression sera lui aussi élevé

Taux de compression

♦ $\tau = \text{volume final} / \text{volume initial}$

$$\left(\tau = \frac{1}{q} \right)$$

plus ce taux de compression est faible,

plus la taille du fichier compressé résultant est faible

♦ $\tau = 1 - \text{volume final} / \text{volume initial}$

$$\left(\tau = 1 - \frac{1}{q} \right)$$

plus ce taux de compression est élevé,

plus la taille du fichier compressé résultant est faible

Coder avec des mots courts (*resp.* longs)
les lettres les plus fréquentes (*resp.* rares)

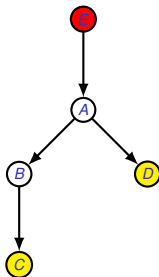
- ♦ Alphabet $\{a, b, c\}$ avec a très fréquente, b moyennement et c rare
- ♦ On peut utiliser $\tau_5(a) = 0$, $\tau_5(b) = 10$ et $\tau_5(c) = 11$
 - ▶ ainsi $\tau_5(aaabaaabbbaaac) = 000100001010000011$, soit 18 bits
 - ▶ un codage de longueur fixe (disons 2 bits/caractère) aurait donné 28 bits

Lettre	Fréquence	Lettre	Fréquence
e	12,10	p	2,49
a	7,11	g	1,23
i	6,59	b	1,14
s	6,51	v	1,11
n	6,39	h	1,11
r	6,07	f	1,11
t	5,92	q	0,65
o	5,02	y	0,46
l	4,96	x	0,38
u	4,49	j	0,34
d	3,67	k	0,29
c	3,18	w	0,17
m	2,62	z	0,15

Fréquence des caractères dans Wikipédia en français

Coder avec des mots courts (*resp.* longs)
les lettres les plus fréquentes (*resp.* rares)

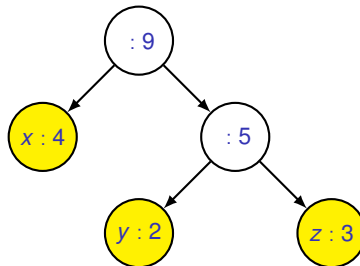
- ♦ Comment obtenir un codage à partir des fréquences des lettres ?
- ♦ On construit un **arbre** : c'est une structure constituée de nœuds
 - ▶ un **nœud** permet de désigner d'autres nœuds
 - ▶ un nœud qui ne désigne rien est une **feuille**
 - ▶ un nœud non désigné est appelé **racine**



- ♦ A, B, C, D, E : nœuds
- ♦ E désigne A , A désigne B et D , B désigne C
- ♦ E : racine
- ♦ C, D : feuilles

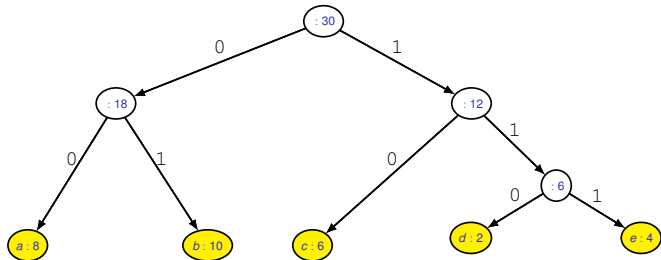
Coder avec des mots courts (*resp.* longs)
les lettres les plus fréquentes (*resp.* rares)

- ◆ Comment obtenir un codage à partir des fréquences des lettres ?
- ◆ On construit un arbre, ici de sorte que
 - ▶ chaque feuille porte une lettre et sa fréquence associée
(ou son nombre d'occurrences)
 - ▶ chaque nœud porte la somme des fréquences des nœuds qu'il désigne



Coder avec des mots courts (*resp.* longs)
les lettres les plus fréquentes (*resp.* rares)

- ◆ Comment obtenir un codage à partir des fréquences des lettres ?
- ◆ On construit un arbre de Huffman, selon l'algorithme :
 - ▶ on part de la **forêt** des arbres réduits à de simples feuilles (portant chacune une lettre pondérée par sa fréquence)
 - ▶ tant qu'il reste au moins deux arbres (dans la forêt), on sélectionne deux arbres dont les fréquences des racines sont **les plus petites** on construit un arbre dont le nœud racine désigne les deux arbres sélectionnés et dont la fréquence est simplement la somme des fréquences



- ▶ on étiquette chaque paire de branche l'une par 0 (gauche) et l'autre par 1 (droite)

Reprinted with corrections from *The Bell System Technical Journal*,
Vol. 27, pp. 379–423, 623–656, July, October, 1948.

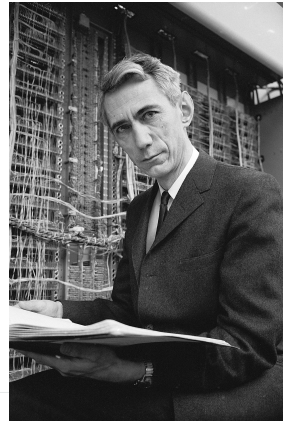
A Mathematical Theory of Communication

By C. E. SHANNON

INTRODUCTION

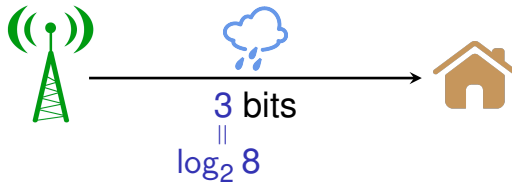
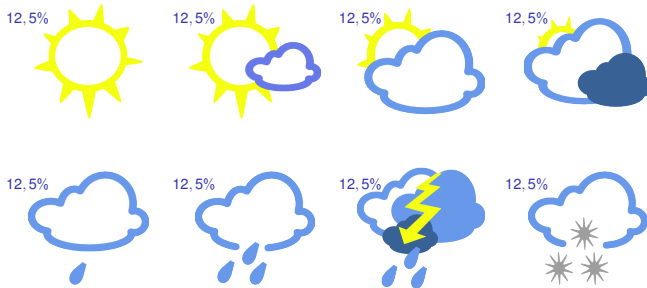
THE recent development of various methods of modulation such as PCM and PPM which exchange bandwidth for signal-to-noise ratio has intensified the interest in a general theory of communication. A basis for such a theory is contained in the important papers of Nyquist¹ and Hartley² on this subject. In the present paper we will extend the theory to include a number of new factors, in particular the effect of noise in the channel, and the savings possible due to the statistical structure of the original message and due to the nature of the final destination of the information.

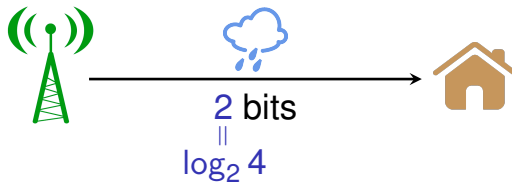
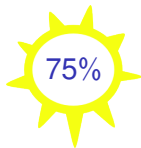
The fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point. Frequently the messages have *meaning*; that is they refer to or are correlated according to some system with certain physical or conceptual entities. These semantic aspects of communication are irrelevant to the engineering problem. The significant aspect is that the actual message is one *selected from a set* of possible messages. The system must be designed to operate for each possible selection, not just the one which will actually be chosen since this is unknown at the time of design.

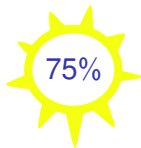


1 bit \leftrightarrow incertitude divisée par 2



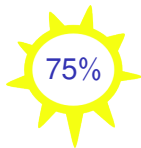






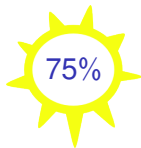
2 bits

$$\begin{array}{c} \parallel \\ \log_2 4 = \log_2 \frac{1}{0,25} \end{array}$$



0,415 bits

$$\approx \log_2 \frac{4}{3} = \log_2 \frac{1}{0,75}$$

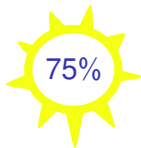



en moyenne ?


0,811 bits



$$0,75 \log_2 \frac{1}{0,75} + 0,25 \log_2 \frac{1}{0,25}$$

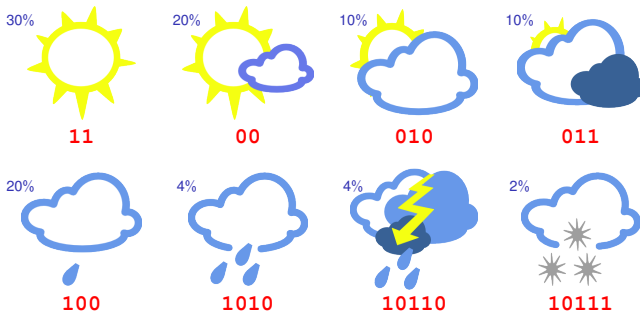




$$H(p) = \sum p_i \log_2 \frac{1}{p_i}$$
entropie


$0,75 \log_2 \frac{1}{0,75} + 0,25 \log_2 \frac{1}{0,25}$

$0,811 \text{ bits}$



$$H(p) = \sum p_i \log_2 \frac{1}{p_i}$$

entropie

$$0,30 \log_2 \frac{1}{0,30} + \dots + 0,02 \log_2 \frac{1}{0,02} \approx 2,599$$

longueur moyenne

$$0,30 \times 2 + \dots + 0,02 \times 5 = 2.660$$