ODO formulaire

Charline DANTINNE

1 Quantité d'information

Pour calculer la quantité d'information contenue est donnée par :

$$\log_2\left(\frac{1}{p}\right)$$

$\Rightarrow$ 1 bit représente donc la quantité d'info permettant de distinguer fiablement 2 valeurs équiprobables.

- octet (byte, B) représente 8 bits d'info
- nibble $= \frac{1}{2}$ octet représente 4 bits d'info
- les préfixes
  K(kilo) $\rightarrow 2^{10}$ ou 10^3

  M(mega) $\rightarrow 2^{20}$ ou 10^6

  G(giga) $\rightarrow 2^{30}$ ou 10^9

  T(tera) $\rightarrow 2^{40}$ ou 10^{12}

  P(peta) $\rightarrow 2^{50}$ ou 10^{15}

  ex : La capacité d'un disque dur de 4TB n'est en réalité que :

$$\frac{4.10^{12}}{2^{40}} \approx 3,64\ TB$$

2 Représentation par valeur non signée

La représentation des nombres se calcule grâce aux propriétés suivantes :

1.

$$\text{bit de poids faible} = 0 \rightarrow pair \quad (1)$$
$$= 1 \rightarrow impair \quad (2)$$

2. en retirant le bit de poids faible, on obtient une représentaion de $\lfloor \frac{\nu}{2} \rfloor$

Ex : on a donc l'algorithme suivant (générant d'abord le bit de poids faibles) :

1. Si ν est pair, afficher 0. Sinon afficher 1
2. Remplacer ν par $\lfloor \frac{\nu}{2} \rfloor$
3. Si $\nu \neq 0$, recommencer à l'étape 1

Représentation sur l'intervalle $[0, ..., 2^n - 1]$

3 Représentation hexadécimale

Cette représentation utilise une base de 16. Elle est très lisible et facile à convertir vers et depuis une notation binaire. (c'est pourquoi elle est très utilisée en informatique)

Un chifre hexadécimal peut prendre 16 valeurs :
0 ; 1 ; 2 ; 3 ; 4 ; 5 ; 6 ; 7 ; 8 ; 9 ; A(10) ; B(11) ; C(12) ; D(13) ; E(14) ; F(15)

ex : $4D2 = 4 \times 16^2 + 13 \times 16^1 + 2 \times 16^0$

Hexadécimal	Binaire	Hexadécimal	Binaire
0	0000	8	1000
1	0001	9	1001
2	0010	*A*	1010
3	0011	*B*	1011
4	0100	*C*	1100
5	0101	*D*	1101
6	0110	*E*	1110
7	0111	*F*	1111

FIGURE 1 – Table de conversion hexadécimal ↔ binaire

Pour préciser qu'on utilise la base hexadécimal, on peut utliser
— l'indice : $xxxx_{16}$
— le suffixe : $xxxx_h$
— le préfixe : $0{\times}xxxx$

4 Représentation des nombres entiers signés

Le bit de poids fort est ici le bit de *signe* :
— **0** pour les nombres **positifs**
— **1** pour les nombres **négatifs**

La représentation des nombres positifs est tjs identique à leur représentation binaire non signée de même taille.

Pour la représentation des nombres négatifs, on représente la valeur absolue du nombre et on ajoute ensuite le bit de signe.

Selon ce procédé, le nombre ν représenté par les bits $b_{n-1}b_{n-2}...b_0$ est :

$$\nu = (1 - 2b_{n-1}) \sum_{i=0}^{n-2} 2^i b_i$$

⇒ l'ensemble des valeurs représentables est donc l'intervalle :

$$\left[-2^{n-1} + 1, ..., 2^{n-1} - 1 \right]$$

Mais, le nombre zéro possède 2 représentations $\boxed{00...0}$ ou $\boxed{10....0}$

5 Représentation par complément à 1

Le bit de poids fort est aussi le bit de *signe* :
— **0** pour les nombres **positifs**
— **1** pour les nombres **négatifs**

Pour obtenir la représentation d'un nombre en complément à 1, on représente la valeur absolue du nombre en binaire non signée. Ensuite, on complémente chaque bit, càd, on remplace les 1 par des 0 et les 0 par des 1

Selon ce procédé, le nombre ν représenté par les bits $b_{n-1}b_{n-2}...b_0$ est :

$$\nu = (1 - 2^n)b_{n-1} + \sum_{i=0}^{n-1} 2^i b_i$$

$\Rightarrow$ l'ensemble des valeurs représentables est donc l'intervalle :

$$\left[-2^{n-1} + 1, ..., 2^{n-1} - 1 \right]$$

Le nombre zéro possède de nouveau 2 représentations $\boxed{00...0}$ (zéro positif) et $\boxed{11...1}$ (zéro négatif)

6 Représentation par complément à 2

Le bit de poids fort est aussi le bit de *signe* :
— **0** pour les nombres **positifs**
— **1** pour les nombres **négatifs**

La représentation d'un nombre par complément à 2 est la représentation du nombre à complément à 1 à laquelle on ajoute 1
Selon ce procédé, le nombre ν représenté par les bits $b_{n-1}b_{n-2}...b_0$ est :

$$\nu = -2^n \ b_{n-1} + \sum_{i=0}^{n-1} 2^i b_i$$

$\Rightarrow$ l'ensemble des valeurs représentables est donc l'intervalle :

$$\left[-2^{n-1}, ..., 2^{n-1} - 1 \right]$$

Le zéro possède ici une seule représentation $\boxed{00...0}$

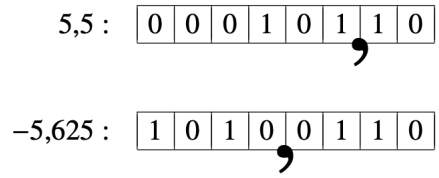
Bits	Non signée	Valeur signée	Compl. à un	Compl. à deux
0000	0	0	0	0
0001	1	1	1	1
0010	2	2	2	2
0011	3	3	3	3
0100	4	4	4	4
0101	5	5	5	5
0110	6	6	6	6
0111	7	7	7	7
1000	8	0	−7	−8
1001	9	−1	−6	−7
1010	10	−2	−5	−6
1011	11	−3	−4	−5
1100	12	−4	−3	−4
1101	13	−5	−2	−3
1110	14	−6	−1	−2
1111	15	−7	0	−1

FIGURE 2 – Tableau récap

7 Représentation en virgule fixe

On introduit un séparateur entre une partie entière et une partie fractionnaire, à une position fixée.

S'il y a k bits après le séparateur, alors le nombre représenté vaut $\frac{1}{2^k}$ fois le nombre entier représenté par la même suite de bits.



Rem : Attention, lorsqu'on passe de la représentation non signée au complément à 2 et qu'on ajoute 1, c'est 1 au bit de poids faible.

8 Représentation en virgule flottante

Permet de corriger le fait que la virgule fixe n'est pas adaptée aux applications où la grandeur des nombres représentés est très variable.

$\nu = m \times r^e$ où :

— **r** est la base

— **m** est la mantisse (en virgule fixe)

— **e** est l'exposant (entier)

8.1 Standard IEEE 754

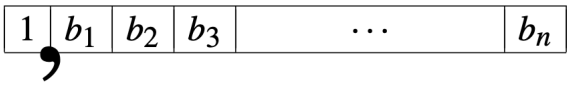
— **Simple précisions** :
1 bit de signe : 0 pour les nombres positifs et 1 pour les nombres négatifs.
8 bits pour l'exposant
23 bits pour la mantisse
exp = e + 127 ⇒ l'ensemble des exposants représentables est $[-127, ..., 128]$

— **Double précisions** :
1 bit de signe : 0 pour les nombres positifs et 1 pour les nombres négatifs.
11 bits pour l'exposant
52 bits pour la mantisse
exp = e + 1023 ⇒ l'ensemble des exposants représentables est $[-1023, ..., 1024]$

Encodage de la mantisse

1. **L'exposant n'est pas égal à la valeur extrême** (-127 ou 128 pour simple précision, -1023 ou 1024 pour la double précision) → la matrice est alors normalisée.
Dans ce cas, la mantisse m est représentée par $b_1 b_2 ... b_n$ (avec n = 23 pour la simple précision est n = 52 pour la double précision) et :

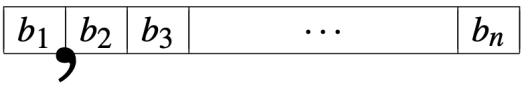
$$|m| = 1 + \sum_{i=1}^{n} 2^{-i} b_i$$



Pour une matrice normalisée : $1 \leq |m| < 2$ (ex p.72 des slides)

2. **L'exposant est égal à sa valeur minimal** (-127 pour simple précision, -1023 pour la double précision) → la matrice est alors dénormalisée.
Dans ce cas, la mantisse m est représentée par $b_1 b_2 ... b_n$:

$$|m| = \sum_{i=1}^{n} 2^{-i+1} b_i$$



Pour une matrice dénormalisée : $0 \leq |m| < 2$
Rem : la représentation de l'exposant -127 est représenté par 0000 0000 (ex p.74 des slides).
Les mantisses dénormalisées permettent de représenter des nombres plus petits (en valeur absolue) qu'avec une mantisse normalisée mais ça engendre une diminution de la précision. ⇒ Il existe 2 représentations de zéros :

00...00 (zéro positif) ET 100...0 (zéro négatif)

(ce ne sont pas de vrais zéros mais l'arrondi positif ou négatif de nombres très très petits)

3. **L'exposant est égal à sa valeur maximale** (128 pour simple précision, 1024 pour la double précision) Cette situation sert à encoder des valeurs spéciales, résultats ne correspondant pas à des réels :

• Tous les bits de la mantisse sont égaux à 0 → dépassement :- bit de signe = 0 : vers les valeurs positives ($+\infty$)- bit de signe = 1 : vers les valeurs négatives ($-\infty$)

• au moins un bit de la mantisse est égal à 1 → valeur indéfinie : NaN (Not a Number)

9 Représentation de textes

Code ASCII

• la caractère est encodé à l'aide de 7 bits d'info. On attribue à chaque symbole un code dans $[0, ..., 127]$

20		30	0	40	@	50	P	60	‘	70	p
21	!	31	1	41	A	51	Q	61	a	71	q
22	”	32	2	42	B	52	R	62	b	72	r
23	#	33	3	43	C	53	S	63	c	73	s
24	$	34	4	44	D	54	T	64	d	74	t
25	%	35	5	45	E	55	U	65	e	75	u
26	&	36	6	46	F	56	V	66	f	76	v
27	’	37	7	47	G	57	W	67	g	77	w
28	(	38	8	48	H	58	X	68	h	78	x
29	)	39	9	49	I	59	Y	69	i	79	y
2A	*	3A	:	4A	J	5A	Z	6A	j	7A	z
2B	+	3B	;	4B	K	5B	[	6B	k	7B	{
2C	,	3C	<	4C	L	5C	\	6C	l	7C	|
2D	-	3D	=	4D	M	5D	]	6D	m	7D	}
2E	.	3E	>	4E	N	5E	^	6E	n	7E	~
2F	/	3F	?	4F	O	5F	_	6F	o		

FIGURE 3 – table ASCII

9.1 Le standard ISO 8859-1

Représentation sur 8 bits → 128 caractères de plus sont représentables.
- les 128 premiers caractères - bit de poids fort = 0 - coïncident avec le code ASCII
- les 128 caractères supplémentaires diffèrent pour chaque variante standard

9.2 Unicode

Unifie la représentation de tous les systèmes d'écriture actuels.
Le code appartient à l'intervalle $[0, 0x10FFFF]$ → représentation sur 21 bits.
Noté : U+k, avec k écrit en hexadécimal

Compression UTF-8 Si $k \in [0, 0x7F] \rightarrow 7bits$, le caractère est représenté sur 1 octet :

$$\boxed{0\mathrm{b}_6 b_5 ... b_0}$$

Si $k \in [0x80, 0x7FF] \rightarrow 11bits$, le caractère est représenté sur 2 octets :

$$\boxed{110\mathrm{b}_{10} b_9 ... b_6} \; \boxed{10\mathrm{b}_5 b_4 ... b_0}$$

Quand on commence un octet par "10", c'est un octet de continuation.

Si $k \in [0x800, 0xFFFF] \rightarrow 11bits$, le caractère est représenté sur 3 octets :

$$\boxed{1110\mathrm{b}_{15} b_{14} ... b_{11}} \; \boxed{10\mathrm{b}_{11} b_{10} ... b_6} \; \boxed{10\mathrm{b}_5 b_4 ... b_0}$$

Si $k \in [0x10000, 0x10FFFF] \rightarrow 11bits$, le caractère est représenté sur 4 octets :

$$\boxed{11110\mathrm{b}_{20} b_{19} b_{18}} \; \boxed{10\mathrm{b}_{17} b_{16} ... b_{12}} \; \boxed{10\mathrm{b}_{11} b_{10} ... b_6} \; \boxed{10\mathrm{b}_5 b_4 ... b_0}$$

Exemple: Pour U+20AC ("€"), on a k = 0x20AC ∈ [0x800, 0xFFFF]. En binaire, 0x20AC s'écrit 0010 0000 1010 1100. Ce symbole est donc représenté par les trois octets

$$\boxed{1110\,0010} \; \boxed{1000\,0010} \; \boxed{1010\,1100},$$

c'est-à-dire E2 82 AC en hexadécimal.