CHAPITRE 5 Représentation des données

1 Représentation des entiers relatifs

Rappel. Il est possible de représenter les **entiers naturels** (les nombres $n \in \mathbb{N}$) à l'aide de leur écriture en base deux.
Par exemple, $\overline{10110100}^2$ représente le nombre : .....
Avec cette représentation, le bit le plus à gauche est appelé le bit de poids fort, le bit le plus à droite est appelé le bit de poids faible.

Définition 1. Soit N un entier naturel. On appelle **capacité** le nombre de nombres binaires qu'il est possible de représenter sur N bits.

Exemple 1. Avec $N = 8$ bits il est possible de coder 256 valeurs (par exemple les entiers positifs de 0 à 255). Avec $N = 16$ bits il est possibles de coder 65 536 valeurs.

Propriété 1. *Soit N un entier naturel.*
*Soient $n_1 \in \mathbb{N}$ et $n_2 \in \mathbb{N}$ deux entiers pouvant s'écrire sur N bits. On ajoute bit à bit les deux nombres en reportant les retenues éventuelles. Si l'addition du bit de poids fort génère une retenue, on dit qu'il y a **dépassement de capacité**.*

Exemple 2. On additionne $n_1 = 42$ et $n_2 = 14$ écrits sur 8 bits.

.....
.....
.....
.....
.....

On additionne $n_1 = 5$ et $n_2 = 6$ écrits sur 8 bits.

.....
.....
.....
.....
.....

Remarque. Pour soustraire deux nombres, on utilise la même méthode, car pour tout entiers a et b on a : $a - b = a + (-b)$.
Il faut donc trouver une manière de représenter $-b$, à savoir les entiers relatifs.

Définition 2. Soit $n \in \mathbb{Z}$. On note $\overline{b_N \dots b_1 b_0}$ l'écriture en base 2 de $|n|$ écrit sur N bits.
Le **complément à 2** $\overline{c_N \dots c_1 c_0}$ de n est défini de la manière suivante :

- Si n est positif, $n = |n|$: dans ce cas il s'agit de l'écriture en base deux usuelle de n.
  On a alors : $\overline{c_N \dots c_1 c_0} = \overline{b_N \dots b_1 b_0}$ et le bit de poids fort est toujours 0.
- Si n est négatif, $n = -|n|$:
  - on inverse les bits de l'écriture binaire de n ;
  - on ajoute 1 au résultat (les dépassements de capacité sont ignorés).

  Dans ce cas on a alors : $\overline{c_N \dots c_1 c_0} = \overline{\sim b_N \dots \sim b_1 \sim b_0} + 1$, où $\sim 0 = 1$ et $\sim 1 = 0$ et le bit de poids fort est toujours 1.

Exemple 3. On représente $n = -5$ écrit sur $N = 8$ bits à l'aide du complément à 2.

.....

.....

.....

.....

.....

On représente $n = -4$ écrit sur $N = 4$ bits à l'aide du complément à 2.

.....

.....

.....

.....

.....

Propriété 2. *Soit N un entier naturel.*
La représentation binaire en complément à 2 sur N bits permet de représenter les 2^N entiers $n \in [-2^{N-1}; 2^{N-1} - 1]$.

Exemple 4. Sur $N = 3$ bits, on peut représenter tous les entiers entre $-2^{-2} = -4$ et $2^2 - 1 = 3$.

Représentation en complément à 2	000	001	010	011	100	101	110	111
Entier n en base 10	0	1	2	3	-4	-3	-2	-1

Propriété 3. *Soit $n = \overline{b_N \dots b_1 b_0}$ un entier positif écrit sur N bits, et $-n = \overline{c_N \dots c_1 c_0}$ son complément à 2. Alors en ajoutant bit à bit et en ignorant les dépassements de capacité on a toujours :*

$$n + (-n) = \underbrace{\overline{0 \dots 0}}_{N \textit{ fois}}$$

Exemple 5. Effectuer le calcul $42 - 14$ à l'aide de la méthode du complément à 2 en écrivant les entiers sur $N = 8$ bits.

.....

.....

.....

.....

.....

.....

.....

.....

Effectuer le calcul $-4 - 3$ à l'aide de la méthode du complément à 2 en écrivant les entiers sur $N = 3$ bits.

.....

.....

.....

.....

.....

Comment corriger ce problème ? .....

.....

2 Représentation des nombres réels

Définition 3. On appelle **partie fractionnaire** de $x \in \mathbb{R}$ le nombre $f \in [0;1[$ tel que $x - f \in \mathbb{Z}$. On appelle **partie entière** de x le nombre $x - f$.

Définition 4. On représente les nombres $f \in [0;1[$ en binaire en les décomposant en somme de puissances *inverses* de 2. Le bit immédiatement après la virgule correspond à 2^{-1}, puis 2^{-2} etc. Pour convertir une partie fractionnaire écrite en base 10 en base 2 :

- On multiplie la partie fractionnaire par 2.
- Extraire la partie entière du résultat : c'est le bit suivant dans la représentation en base 2.
- Continuer jusqu'à ce que la partie fractionnaire soit nulle.

Exemple 6. Convertir $\overline{0,011}^2$ en base 10.

.....
.....
.....
.....

Convertir $\frac{1}{3}$ en base 2.

.....
.....
.....
.....

Convertir $\overline{0,8125}^{10}$ en base 2.

.....
.....
.....
.....

Convertir $0,1$ en base 2.

.....
.....
.....
.....

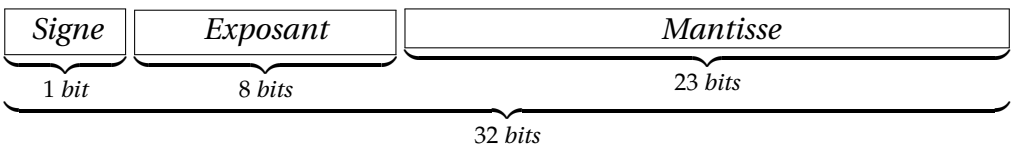
Remarque. Pour représenter les nombres réels (à approximation près), on utilise un système similaire à la notation scientifique. Par exemple, le nombre d'Avogadro est $6,0221 \times 10^{23}$ et la masse du proton est $9,1094 \times 10^{-31}$. Dans les deux cas, on a utilisé 5 chiffres significatifs et deux chiffres pour l'exposant (en plus du signe).

Propriété 4. *On utilise en binaire la décomposition suivante :*

$$x = \textit{signe} \times (1 + \textit{mantisse}) \times 2^{\textit{exposant}} \qquad \textit{avec } 0 \leq \textit{mantisse} < 1$$

Les nombres flottants (précision simple, norme IEE 754) stockés sur 32 bits vérifient :

- *1 bit pour le signe ;*
- *8 bits pour l'exposant ;*
- *23 bits pour la mantisse représentée en base 2.*



L'exposant est un entier compris entre -126 *et* 127*, mais les valeurs correspondant à* -127 *et* 128 *sont réservés respectivement pour coder d'une façon particulière les nombres très proches de* 0*,* $+\infty$*,* $-\infty$ *et* NaN*. (Not a Number, le résultat de* $+\infty - \infty$ *par exemple.)*

Exemple 7. La constante d'Avogadro $6,0221 \times 10^{23}$ s'encode ainsi en :

`01100110111111110000101110111101`

Source. https://www.h-schmidt.net/FloatConverter/IEEE754.html

Remarque. Ceci explique :

Code python

```
print(0.1 + 0.2)
```

Résultat

```
0.30000000000000004
```

Pour comparer des nombres flottants, on n'utilise **jamais** l'opérateur ==. On utilise plutôt une marge d'erreur. Par exemple à 10^{-3} près :

Code python

```
print(abs(0.1 + 0.2 - 0.3) <= 10**(-3))
```

Résultat

```
True
```

3 Représentation des caractères

Définition 5. L'American Standard Code for Information Interchange (Code américain normalisé pour l'échange d'information : ASCII) est une norme informatique d'encodage de caractères. Elle contient 128 points de code et permet d'encoder les chiffres arabes de 0 à 9, les 26 lettres de l'alphabet latin en minuscules et en capitales, des symboles mathématiques et de ponctuation, ainsi que des caractères spéciaux.

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[END OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Remarque. Dans ce code, seules 128 valeurs sont utilisés (impossible d'encoder la lettre "ç" par exemple !). Le bit de poids fort d'un octet représentant un caractère est donc toujours 0.

Exemple 8. Quel texte représente les trois octets suivants : `01001110 01010011 01001001` ?

.....

.....

.....

Définition 6. La norme Unicode attribue un identifiant numérique, appelé point de code, différent à chacun des milliers de caractères nécessaires à la transcription des différentes langues mondiales, existantes ou non, on retrouve par exemple le klingon et l'elfique et différents éléments tels que les émoticônes.
Cette norme ne précise pas sous quelle forme cet identifiant doit être encodé par le système informatique. Il existe donc plusieurs normes d'encodages différentes en fonction des besoins, mais chacune a en commun d'associer à chaque caractère le même identifiant numérique.

Définition 7. L'encodage UTF-8 (Unicode Transformation Format) est un code à taille variable destiné à représenter les codes Unicode. Le fonctionnement est le suivant :

- Si le bit de poids fort est à 0, le point de code est codé sur un octet, et correspond à un caractère ASCII.
- Si le bit de poids fort est à 1, le nombre de 1 consécutifs indique le nombre d'octet utilisé pour coder le point de code. Les deux bits de poids forts des octets suivants sont fixés à 10 pour indiquer qu'ils continuent la séquence.

Nombre d'octets	Premier code	Dernier code	Octet 1	Octet 2	Octet 3	Octet 4
1	0000	007F	0xxxxxxx			
2	0080	07FF	110xxxxx	10xxxxxx		
3	0800	FFFF	1110xxxx	10xxxxxx	10xxxxxx	10xxxxxx
4	10000	10FFFF	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

Exemple 9. Le message suivante est codé en UTF-8, combien de caractères sont-ils représentés ? Combien d'entre eux correspondent à un caractère ASCII ?

`0110001 0110101 11100010 10000010 10101100`

.....

.....

.....

.....

Remarque. On peut obtenir en Python le code d'un caractère à partir de la fonction `ord`, et inversement on peut obtenir un caractère à partir de son code à l'aide de la fonction `chr`.
Par exemple :

Code python

```
1 print(ord("a"))
2 print(ord("A"))
3 print(ord("€"))
4 print(chr(200))
5 print(chr(240))
```

Résultat

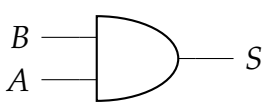
```
97
65
8364
È
ð
```

4 Circuits combinatoires

Définition 8. Un **circuit combinatoire** est un circuit électronique muni de n entrées et de m sorties au format binaire (0/1). Sa spécification se fait à base de fonctions booléennes ou de tables de vérité.
Les **portes logiques** sont des circuits combinatoires élémentaires, qu'il est possible de réaliser éléctroniquement à l'aide de transistors.

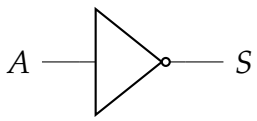
Propriété 5. *Si C est un circuit combinatoire quelconque, alors il est possible de le construire uniquement à l'aide des portes AND et NOT.*

Porte AND



A	B	S
0	0	0
0	1	0
1	0	0
1	1	1

Porte NOT



A	S
1	0
0	1