Mathématiques Discrètes

Chapitre 4
Numération

Leo Donati Noëlle Stolfi

Université de Nice Sophia Antipolis
IUT Nice Côte d'Azur
DUT Informatique

2015-2016

Chapitre 4 : Numération

Ecriture positionnelle

Base et chiffres

Soit $B \geq 2$ la base du système de numération.
En base B les chiffres sont B symboles différents $0, 1, 2, \ldots, B-1$.

Écriture positionnelle en base B de x

Tout réel x positif s'écrit en base B comme une suite (éventuellement infinie)

$$x = (c_k c_{k-1} c_{k-2} \cdots c_1 c_0, c_{-1} c_{-2} \cdots)_B$$

où les c_i sont des chiffres en base B, si et seulement si on a :

$$x = \sum_{i \leq k} c_i B^i$$

Partie entière et fractionnaire

Définition

La partie entière de x, est la suite des chiffres d'indice positif ou nul.

$$\text{int}(x) = (c_k c_{k-1} c_{k-2} \cdots c_1 c_0)_B$$

La partie fractionnaire de x, est la suite des chiffres d'indice négatif.

$$\text{frac}(x) = (0, c_{-1} c_{-2} \cdots)_B$$

Remarque

- $\forall x \in \mathbb{R}^+,\ \text{int}(x) \in \mathbb{N}$
- $\forall x \in \mathbb{R}^+,\ 0 \leq \text{frac}(x) < 1$
- $\forall x \in \mathbb{R}^+,\ x = \text{int}(x) + \text{frac}(x)$

Conventions d'écriture

Remarque

Dans l'écriture positionnelle de x on observe certaines conventions d'écriture :

- le premier chiffre c_k doit être non nul sauf si c'est c_0.
- s'il existe un entier négatif ou nul m tel que $c_m \neq 0$ et $\forall i < m,\ c_i = 0$ alors on n'écrit pas les 0 qui suivent, c'est à dire que l'on écrira

$$x = (c_k c_{k-1} c_{k-2} \cdots c_1 c_0, c_{-1} c_{-2} \cdots c_m)_B$$

On dira dans ce cas que x admet une partie fractionnaire limitée. Si par contre un tel entier m n'existe pas, alors x admet une partie fractionnaire illimitée.

Base 10

Rappel

En base 10, chaque chiffre, selon sa position, est multiplié par une puissance de 10.

Exemple

Le nombre $N = (4275,306)_{10}$ représente

$$4 \cdot 10^3 + 2 \cdot 10^2 + 7 \cdot 10^1 + 5 \cdot 10^0 + 3 \cdot 10^{-1} + 6 \cdot 10^{-3}$$

Et l'on a

- $\text{int}(N) = (4275)_{10}$
- $\text{frac}(N) = (0,306)_{10}$.

Base 2

Bits

Les chiffres 0 et 1 du système de numération en base 2, dit système binaire, sont appelés bits à partir des mots anglais *binary digits*.

Ecriture binaire

$$\begin{aligned}(11001,1)_2 &= 1\cdot 2^4+1\cdot 2^3+0\cdot 2^2+0\cdot 2^1+1\cdot 2^0+1\cdot 2^{-1}\\ &= 16+8+1+0,5\\ &= 25,5_{10}\end{aligned}$$

Base 3

Système ternaire

Les chiffres du système de numération en base 3 sont $0, 1, 2$.

Exemple

On écrit alors tout nombre réel comme la somme de 0, 1 ou 2 puissances de 3.

$$(25)_{10} = 2 \cdot 9 + 2 \cdot 3 + 1 = (221)_3$$

Viceversa

$$\begin{aligned} (2102,2)_3 &= 2 \cdot 3^3 + 1 \cdot 3^2 + 0 \cdot 3^1 + 1 \cdot 3^0 + 2 \cdot 3^{-1} \\ &= 2 \cdot 27 + 9 + 2 + 0,6666\ldots \\ &= 65,6666\ldots_{10} \end{aligned}$$

Base 4

Remarque

Comme $4 = 2^2$ tous les chiffres de la base 4 s'écrivent en binaire avec 2 bits :

$$\begin{aligned} 00_2 &= 0_4 \\ 01_2 &= 1_4 \\ 10_2 &= 2_4 \\ 11_2 &= 3_4 \end{aligned}$$

Donc la conversion entre base 2 et 4 est aisée :

$$(57,5)_{10} = (111001,1)_2 = (321,2)_4$$

Base 8

Système octal

Les chiffres du système octal sont 0, 1, 2, 3, 4, 5, 6, 7.
Chacun s'écrit avec 3 bits en binaire.

Exemple

$$\begin{aligned}(74,1)_8 &= 7\cdot 8+4+8^{-1} = (60,125)_{10}\\ &= (111\ 100,001)_2\end{aligned}$$

Base 16

Système hexadécimal

Les chiffres du système hexadécimal sont 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.
Chacun s'écrit avec 4 bits en binaire.

Exemples

1. $F1_{16} = 15 * 16 + 1 = (81)_{10}$
2. $1,A_{16} = 1 + 10 * 16^{-1} = 1 + \frac{10}{16} = \frac{26}{16}$
3. $F1_{16} = (1111\ 0001)_2$

Chapitre 4 : Numération

Conversion vers la base 10

Méthode

Il suffit de multiplier chaque chiffre de la base B par la bonne puissance de B et faire la somme.

Exemples

- depuis la base 5 :
  $(432,1)_5 = 4 \cdot 5^2 + 3 \cdot 5 + 2 + 1 \cdot 5^{-1} = (117,2)_{10}$
- depuis la base 11 :
  $(AAA)_{11} = 10 \cdot 11^2 + 10 \cdot 11 + 10 = 1210 + 110 + 10 = (1330)_{10}$

Conversion depuis la base 10

Conversion de la partie entière

Pour convertir un nombre entier N de la base 10 vers la base B :

1. on fait la division entière de N par B : $N = B \cdot q + r$
2. r est le premier chiffre de l'écriture de N en base B
3. on recommence avec q à la place de N
4. on stoppe quand $q = 0$

Exemple binaire

Ecriture binaire de 109

$$\begin{aligned} 109 &= 2 \times 54 + 1 \\ 54 &= 2 \times 27 + 0 \\ 27 &= 2 \times 13 + 1 \\ 13 &= 2 \times 6 + 1 \\ 6 &= 2 \times 3 + 0 \\ 3 &= 2 \times 1 + 1 \\ 1 &= 2 \times 0 + 1 \end{aligned}$$

Et on trouve $(109)_{10} = (1101101)_2$.

Conversion de la partie fractionnaire

Méthode

Pour convertir un nombre décimal $x < 1$ en base B :

1. on multiplie x par B : $y = x \cdot B$
2. la partie entière de y est le premier chiffre après la virgule de l'écriture de x en base B
3. on recommence avec $\text{frac}(y)$ à la place de x
4. on stoppe quand $\text{frac}(y) = 0$

Exemple binaire

Ecriture binaire de $(0,78125)_{10}$

$$\begin{aligned} 0,78125 \times 2 &= 1,5625 \\ 0,5625 \times 2 &= 1,125 \\ 0,125 \times 2 &= 0,25 \\ 0,25 \times 2 &= 0,5 \\ 0,5 \times 2 &= 1 \end{aligned}$$

Donc $(0,78125)_{10} = (0,11001)_2$.

Exemple en base 5

Ecriture de 64,7 en base 5

- Partie entière

$$\begin{aligned} 64 &= 5 \times 12 + 4 \\ 12 &= 5 \times 2 + 2 \\ 2 &= 5 \times 0 + 2 \end{aligned}$$

- Partie fractionnaire

$$\begin{aligned} 0,7 \times 5 &= 3,5 \\ 0,5 \times 5 &= 2,5 \\ 0,5 \times 5 &= 2,5 \\ \ldots &\quad \ldots \end{aligned}$$

- En définitive

$$(64,7)_{10} = (224,322\ldots)_5$$

Chapitre 4 : Numération

Ecriture en base B des nombres rationnels

Théorème

Tous les nombres rationnels s'écrivent, en base B

- soit avec une écriture fractionnaire finie
- soit avec une écriture fractionnaire infinie mais périodique

Réciproque

Un nombre qui en base B a une écriture fractionnaire finie ou périodique est forcément un nombre rationnel

Corollaire

Un nombre irrationnel a une écriture fractionnaire illimitée et non périodique dans toutes les bases.

Unicité ?

Problème

Y a–t–il unicité dans l'écriture d'un nombre N dans une base B ?

En général, NON

Exemple (en base 10)

On considère le nombre $x = (0,999999999\ldots)_{10}$
Alors, $10 \cdot x = (9,999999999\ldots)_{10}$ donc en soustrayant cette formule à la précédente :

$$10x - x = 9 \Rightarrow 9x = 9 \Rightarrow x = 1$$

Donc en conclusion j'ai deux façons différentes d'écrire le nombre 1.

Chapitre 4 : Numération

Somme

Addition en base B

L'algorithme d'addition usuel, somme chiffre par chiffre de la droite à la gauche en reportant les retenues, fonctionne dans toutes les bases

Il suffit de connaître les tables d'addition.

Tables en base 2, 3 et 4

+	0	1
0	0	1
1	1	10

+	0	1	2
0	0	1	2
1	1	2	10
2	2	10	11

+	0	1	2	3
0	0	1	2	3
1	1	2	3	10
2	2	3	10	11
3	3	10	11	12

Multiplication

Multiplications en base B

L'algorithme de produit usuel, fonctionne dans toutes les bases
Il suffit de connaître les tables de multiplication.

Tables en base 2, 3 et 4

$\times$	0	1
0	0	0
1	0	1

$\times$	0	1	2
0	0	0	0
1	0	1	2
2	0	2	11

$\times$	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	10	12
3	0	3	12	21

Exemple

Exemple de produit en base 2

		1	1	0	0	1
×				1	0	1
		1	1	0	0	1
1	1	0	0	1		
1	1	1	1	1	0	1

Exemple de produit en base 3

		2	1	2	0	2
×				1	0	2
1	0	0	0	1	1	1
2	1	2	0	2		
10	1	2	1	0	1	1

Complément

Définition

Si N est un entier naturel, alors le complément à la base B de N, noté $C_B(N)$ est le nombre positif qu'il faut rajouter à N pour avoir la plus proche puissance de B.
Si N en base B s'écrit avec k chiffres alors

$$C_B(N) = B^k - N$$

Exemples

En base 10, $C_{10}(381) = 619$ car $381 + 619 = 10^3$.
En base 2, $C_2(1011\ 1100) = 0100\ 0100$ car
$1011\ 1100 + 0100\ 0100 = 1\ 0000\ 0000 = 2^8$.

Propriétés

Cas du binaire

En base 2, on calcule le complément à 2 de $N = 1100\ 0100$ en deux étapes :

1. on calcule le complément à 1 : $0 \leftrightarrow 1$

$$C_1(N) = 0011\ 1011$$

2. on ajoute 1 pour obtenir le complément à 2

$$C_2(N) = 0011\ 1011 + 1 = 0011\ 1100$$

Pour tout entier N et toute base B

$$C_B(C_B(N)) = N$$

Soustraction

Soustraction en utilisant le complément

Le complément à la base permet de remplacer une soustraction par l'addition du complémentaire.
Pour faire une soustraction en base 10 on va avoir

$$A - B = A + C_{10}(B) - 10^k$$

où k est le nombre de chiffre utilisé pour écrire A et B.

Soustraction en base 10

Calcul de $A-B$

avec $A = 2389$ et $B = 778$.

1. on calcule le complément à 10 de B sur 4 chiffres

$$C_{10}(0778) = 9222$$

2. on ajoute le complément :

$$A + C_{10}(B) = 2389 + 9222 = 12611$$

3. on enlève le 1 qui dépasse.

$$A - B = 12611 - 10^4 = 2611.$$

Soustraction en base 2

En base 2

En base 2, le bénéfice de passer par le complément à 2 est évident :

- le complément à 1 est juste une négation des bits
- la somme est une opération élémentaire
- la méthode du complément à 2 est utilisée pour stocker en mémoire les nombres négatifs : pour mémoriser un nombre entier négatif A, ce qui est écrit en mémoire est en fait $C_2(-A)$.

Chapitre 4 : Numération

Codage en machine

Codage en machine

Dans un ordinateur toutes les données sont codées avec des 0 et des 1 sans forcément être des nombres :

- du texte (encodage ASCII, UTF-8, UTF-16)
- de la musique
- des images
- de la video.

On utilise des nombres binaires (écrits en hexadécimal) pour décrire le contenu d'une zone mémoire, même si ce n'est pas un nombre qui est stocké.
Exemple : les caractères imprimables de la table ASCII vont de 20_{16} à $7E_{16}$

Codage des nombres

Différents codages

Il y a différentes façons d'encoder des nombres en mémoire selon le type de nombre que l'on veut stocker. Les principaux sont les suivants :

- si on a des nombres entiers seulement positif ou nul, on va choisir un encodage comme entier non signé
- si on des entiers positifs ou négatifs on choisit le codage des entiers signés
- si on veut travailler avec des nombres à virgule, on choisit les représentations à virgule flottante

Chaque format à sa syntaxe et existe en plusieurs taille.

Entiers non signés

Codage

Si l'on doit représenter un entier N positif ou nul dans une mémoire de k bits, on code N en l'écrivant en binaire dans la mémoire (s'il rentre).
Le plus grand entier que l'on peut écrire avec k bits est $2^k - 1$.

Exemples

- sur 8 bits : $0 \leq N \leq 255$
- sur 16 bits : $0 \leq N \leq 2^{16} - 1$
- sur 32 bits : $0 \leq N \leq 2^{32} - 1$

Entiers signés

Codage

Si l'entier N peut être positif ou négatif, on doit utiliser un des bits pour coder l'information sur le signe :

- Si $N \geq 0$:
  - le premier bit est 0 ;
  - on écrit N en binaire à l'aide des $k-1$ bits restants.
- Si $N < 0$:
  - le premier bit est 1 ;
  - on écrit le complément à 2 de $|N|$ en binaire à l'aide des $k-1$ bits restants.

Exemples

Ecriture de 5 et -5 sur 8 bits

- si $N = 5 > 0$, alors $N = 101_2$ et donc la représentation sera 0000 0101 sur 8 bits.
- si $N = -5 < 0$, alors $|N| = 5 = 101_2$ et $C_1(|N|) = 111\ 1010$ et $C_2(|N|) = 111\ 1011_2$. Donc -5 est représenté par 1111 1011.

Remarque

Avantage : $5 + (-5) = 0000\ 0101 + 111\ 11011 = 1\ 0000\ 0000$ qui est 0 sur 8 bits.

Quelles limites ?

Déséquilibre

Quels sont alors les nombres qui peuvent être écrits avec k bits en mode signé ?

- si $N \geq 0$ alors $0 \leq N \leq 2^{k-1} - 1$
- si $N < 0$ alors $-2^{k-1} \leq N < 0$

Au total : $2^{k-1} - 1$ nombres positifs, 1 nombre nul et 2^{k-1} nombres négatifs. Ce qui donne 2^k possibilités.

Cas usuels

- sur 8 bits : $-128 \leq N \leq +127$
- sur 16 bits : $-2^{15} \leq N \leq 2^{15} - 1$
- sur 32 bits : $-2^{31} \leq N \leq 2^{31} - 1$

Erreur de dépassement

Overflow

Si l'on dépasse ces intervalles on obtient des erreurs de calcul. Par exemple avec des entiers signés sur un octet :

$$\begin{aligned} 127+1 &= 0111\ 1111+0000\ 0001 \\ &= 1000\ 0000 \\ &= -128 \end{aligned}$$

car $C_2(1000\ 0000) = 0111\ 1111+1 = 1000\ 0000 = 128$.

Séquence sur 8 bits

$$
\begin{aligned}
127 &= 0111\ 1111 \\
\dots &= \dots \\
3 &= 0000\ 0011 \\
2 &= 0000\ 0010 \\
1 &= 0000\ 0001 \\
0 &= 0000\ 0000 \\
-1 &= 1111\ 1111 \\
-2 &= 1111\ 1110 \\
-3 &= 1111\ 1101 \\
\dots &= \dots \\
-128 &= 1000\ 0000
\end{aligned}
$$

Nombres à virgule flottante

- Codage le plus utilisé pour stocker des nombres qui ne sont pas forcément entiers (standard international IEEE 754) ;
- basé sur l'écriture scientifique de x en base 2 :

$$x = \pm 1, c_{-1}c_{-2}c_{-3}c_{-4}\cdots c_{-k} \times 2^e$$

Ecriture s-e-m sur 32 ou 64 bits

- s est le signe sur 1 bit : 0 pour positif et 1 pour négatif.
- e, l'exposant auquel on applique un décalage est écrit en binaire sur 8 bits (*float*) ou 11 bits (double).
- m, la mantisse (privée du 1 initial), $c_{-1}c_{-2}c_{-3}c_{-4}\cdots c_{-k}$ sur k bits ; avec $k = 23$ (float) ou $k = 52$ (double).

Décalage de l'exposant

Principe

Dans l'écriture scientifique d'un nombre, l'exposant peut être positif ou négatif. $\Rightarrow$ il faudrait donc aussi coder le signe de l'exposant

Le choix fait par le standard IEEE754 est

- ajouter un nombre fixe à l'exposant (127 pour un float) : $e' = e + 127$
- coder le nombre obtenu en binaire (sur 8 bits pour un float)

Exemples

- si l'exposant est -3 on code 124
- si l'exposant est $+3$ on code 130
- de ce fait les exposants sont dans l'intervalle de -127 à $+128$

Valeurs spéciales

Plus ou moins infini

Les valeurs $\pm\infty$ sont codées en ne mettant que des 1 dans l'exposant et que des 0 dans la mantisse

NaN

NaN signifie Not A Number et peut être un résultat retourné par certains calculs (comme une division par zéro).
Il est codé en ne mettant que des 1 dans l'exposant, et autre chose que des zéros dans la mantisse.

Plus ou moins zéro

Il y a deux façons d'écrire 0 en tant que flottant, en mettant le signe plus ou moins.

Exemple de float

Exemple de codage

Si on veut coder $-13,75$ comme un nombre flottant. On a :

$$13,75 = 1101,11_2 = 1,10111 \cdot 2^3$$

Donc :

- $s = 1$
- $e = 3 + 127 = 130 = 1000\ 0010_2$
- $m = 101\ 1100\ 0000\ 0000\ 0000\ 0000_2$

Ce qui donne en mémoire :

$$1\ \ 1000\ 0010\ \ 101\ 1100\ 0000\ 0000\ 0000\ 0000 = C1\ 5C\ 00\ 00_{16}$$

Exemple de float – suite

Exemple de décodage

Si un nombre flottant est codé par :

$$45\ 3C\ F0\ 00_{16} = 0100\ 0101\ 0011\ 1100\ 1111\ 0000\ 0000\ 0000_2$$

Alors

- $s = 0$ donc le nombre est positif
- $e = 1000\ 1010_2 = 138_{10}$ donc l'exposant de l'écriture scientifique est $138 - 127 = 11$
- $m = 011\ 1100\ 1111\ 0000\ 0000\ 0000_2$

Ce qui donne le nombre :

$$x = +1,01111001111 \cdot 2^{11} = 1011\ 1100\ 1111_2 = 3023_{10}$$

Arrondi et précision

Précision

- Les nombres flottants en simple précision (32 bits) conservent 23 bits significatifs en plus du 1 avant la virgule. Leur précision en base 10 et de 7 décimales car $\log_{10}(2^{24}) \approx 7,2$.
- Pour la double précision, on a 53 bits de précision, soit 15 décimales.

Arrondi

Si le nombre à coder a une écriture scientifique en base 2 avec une mantisse de plus de 23 bits, on arrondit avec la règle suivante :

- si le $24^{\text{ème}}$ bit est 1, on arrondit par excès
- si le $24^{\text{ème}}$ bit est 0, on arrondit par défaut