MATHÉMATIQUES ET INFORMATIQUE
Sciences
Université de Paris

NUMÉRATION LOGIQUE

C5 : représentation des réels

Nicole VINCENT

MATHÉMATIQUES ET INFORMATIQUE Sciences Université de Paris

Les réels

- Une représentation en virgule fixe :
  - La masse de la terre est de 5 973 600 000 000 000 000 000 000 kg
  - La masse du soleil est de 19 891 000 … 000 kg (26 zéros)
  - La masse d'un électron est de 0,00 … 000 91093822 grammes (27 zéros)
- On utilise plutôt la notation scientifique de la forme a x 10^e, e ∈ ℤ
  - notation scientifique normalisée on a 1≤|a|< 10,
  - notation ingénieur 1 ≤ |a|<10^3 et l'exposant e est multiple de 3

Pour les masses de la terre et du soleil, on écrit 5,9736 10^{24} kg et 1,9891 10^{30} kg
Pour l'électron on a 9,1093822 10^{-31} kg

MATHÉMATIQUES ET INFORMATIQUE Sciences Université de Paris

Ecriture d'un réel en machine

- La taille du mot mémoire influence la précision de la représentation des nombres
- Pour représenter exactement un rationnel r = $\frac{n}{d}$ il faut considérer le numérateur n ∈ ℤ et le dénominateur d ∈ ℕ*, sauf si dans la base choisie, r admet un développement fini
- Un nombre irrationnel x ∈ ℝ \ℚ ne peut jamais être représenté exactement
- Sur un ordinateur, on utilise les nombres à **virgule flottante** de la forme

$$\boxed{x = s\ m\ b^e}$$

  - b est la base
  - s ∈ {-1;+1} est le signe
  - la mantisse m précise les chiffres significatifs
  - l'exposant e donne l'ordre de grandeur.

Exemple en base 10 : -37,5 = -37500 10^{-3} = -0,000375 10^5 = -0,375 10^2

MATHÉMATIQUES ET INFORMATIQUE Sciences Université de Paris

Réels en virgule flottante, base 10

Représentation en virgule flottante **normalisée** x = s m 10^e : x = -0,375 10^{+2}

→ le signe du nombre $s = (-1)^{sm}$, avec s_m ∈ {0, 1}

→ la mantisse m ∈]0; 1[(tous les chiffres significatifs sont à droite de la virgule)

→ le digit de poids fort de la mantisse est différent de zéro, le zéro est donc non représentable

→ l'exposant e est un entier relatif

→ la virgule et la base sont représentées de façon implicite

→ cette représentation du nombre est unique : $(-1)^{sm}$ 0,$d_{-1}d_{-2}$ … d_{-p} 10^e

MATHÉMATIQUES ET INFORMATIQUE Sciences Université de Paris

Exemple en base dix

- On considère une représentation avec
  1. une mantisse de 3 chiffres décimaux
  2. un exposant de 2 chiffres
  3. deux bits de signe

Exemple : 37,5 = +0,375 . 10^{+2}

+	+	0	2	3	7	5

- Les nombres strictement positifs représentables vont de +0,100 10^{-99} à +0,999 10^{+99}
  - $\boxed{\text{+-99100}}$ $\boxed{\text{++99999}}$
- Les nombres strictement négatifs représentables vont de -0,999 10^{+99} à -0,100 10^{-99}
  - $\boxed{\text{-+99999}}$ $\boxed{\text{--99100}}$
- Aucun réel de l'intervalle [-0,100 10^{-99} ; 0,100 10^{-99}] n'est représentable

MATHÉMATIQUES ET INFORMATIQUE Sciences Université de Paris

Problèmes de la représentation en virgule flottante

- On ne peut pas représenter des réels plus grands que M

Une opération ayant comme résultat un tel nombre engendre un dépassement de capacité ou **overflow**

En base 10 avec mantisse sur 2 termes et e ∈ {-1, 0, 1}.
Les nombres 3 et 7 sont représentables : 3 = 0,30 10^1 ; 7 = 0,70 10^1

Mais le résultat de 3 + 7 = 10 = 0,10 10^2 n'est pas représentable, d'où **overflow**

MATHÉMATIQUES ET INFORMATIQUE Sciences Université de Paris

Problèmes de la représentation en virgule flottante

- On ne peut représenter des réels $x \in \mathbb{R}$ pour $0 < x < m$

Une opération ayant comme résultat un tel nombre engendre un souspassement de capacité ou **underflow**

En base 10 avec mantisse sur 2 termes et $e \in \{-1, 0, 1\}$

Les nombres $0{,}010 = 0{,}10\ 10^{-1}$ et $0{,}011 = 0{,}11\ 10^{-1}$ sont représentables

mais leur différence $0{,}011 - 0{,}010 = 0{,}001 = 0{,}10\ 10^{-2}$ ne l'est pas, d'où **underflow**

7

MATHÉMATIQUES ET INFORMATIQUE Sciences Université de Paris

Problèmes de la représentation en virgule flottante

- Dans l'intervalle [m , M] on ne représente qu'un nombre fini de réels
- valeurs non distribuées de façon uniforme
- Une opération ayant comme résultat un nombre x non représentable engendre une **erreur d'arrondi**

→ on doit approximer le "vrai" résultat x par un réel $\tilde{x}$ représentable dans le système virgule flottante

En base 10, p = 2 et $e \in \{-1, 0, 1\}$

$1{,}0 - 0{,}011 = 0{,}989$

n'est pas représentable **arrondi** $0{,}99 = 0{,}99\ 10^0$

$5 + 0{,}09 = 0{,}509\ 10^1$ **arrondi** $0{,}51\ 10^1$

8

MATHÉMATIQUES ET INFORMATIQUE Sciences Université de Paris

Représentation en virgule flottante

- Si on relâche la condition que le digit de plus fort poids de la mantisse soit non nul, on peut représenter

$$0{,}001 = 0{,}01\ 10^{-1}$$

- Ces nombres "sous-normaux" (subnormal) sont utilisés pour représenter des quantités très petites mais non nulles

9

MATHÉMATIQUES ET INFORMATIQUE Sciences Université de Paris

L'addition en virgule flottante

L'opération n'est pas associative

En base 10, p = 2 et $e \in \{-1, 0, 1\}$

$a = -9 \quad b = 9 \quad c = 0{,}011 = 0{,}11\ 10^{-1}$

Calcul de $d = a + b + c$

$(a + b) + c = 0{,}011$

$a + (b + c) = 0$

$b + c = 9{,}011 = 0{,}9011\ 10^1$

non représentable,

arrondi en $0{,}90\ 10^1 = b$

$a + b = 0$

10

MATHÉMATIQUES ET INFORMATIQUE Sciences Université de Paris

Bilan : représentation en virgule flottante

Représentation en virgule flottante en base b : $(-1)^s\ 0{,}d_{-1}d_{-2} \ldots d_{-(p-1)}d_{-p}\ b^e$

$$e_m \leq e \leq e_M,\ d_i \in \{0, 1, \ldots, b - 1\} \text{ et } d_{-1} \neq 0$$

- sous-ensemble fini de $\mathbb{R}$ non stable pour les opérations arithmétiques. Il y aura toujours des overflows, underflows et erreurs d'arrondi
- non répartis de façon uniforme
- Le nombre $\varepsilon = b^{1-p}$ est tel que entre 1 et $1 + \varepsilon$ aucun réel n'est représentable est la **précision machine**
  Sert à majorer les erreurs d'arrondi et d'approximation

11

MATHÉMATIQUES ET INFORMATIQUE Sciences Université de Paris

Réels en virgule flottante en base 2

- Le standard **IEEE 754-2008** fixe, entre autres, comment représenter des nombres flottants en simple (4 octets) et double (8 octets) précision.
  On distingue souvent les formats float et double
- Ce standard fixe aussi : les modes d'arrondi, les calculs avec les nombres flottants, des valeurs particulières, la détection de problèmes (overflow, . . .)
- On étudie ici deux points concernant le format :
  - les exposants biaisés
  - le bit implicite

IEEE = Institute of Electrical and Electronics Engineers (association professionnelle internationale, de droit américain)

12

MATHÉMATIQUES ET INFORMATIQUE Sciences Université de Paris

Réels en virgule flottante en base 2

Représentation avec le format

Signe mantisse	Exposant	Mantisse normalisée
1 bit	4 bits	8 bits

exemple $(0{,}015)_8$

$(0{,}015)_8 = (0{,}000001101)_2 = (0{,}1101)_2 \,.\, (2^{-5})_{10}$

Mantisse positive, donc bit de signe égal à 0

Mantisse normalisée $(0{,}1101)_2$

Exposant $(-5)_{10}$ Codage en complément à deux sur 4 bits

$(5)_{10} = (101)_2$, inverse [1010] , d'où le code CA2 : 1011

Représentation du nombre : 0 1011 11010000

Exposant biaisé

- Un comparateur logique, opérant bit à bit et de gauche à droite, ne peut pas facilement comparer des nombres (difficile en CA2)
- Pour simplifier les comparaisons sur les mots mémoire, les exposants signés sont codés grâce à un **décalage**
- En code CA2, sur k bits, on représente les entiers signés $-2^{k-1} \leq n \leq 2^{k-1} - 1$
- En ajoutant 2^{k-1}, on translate l'intervalle sur $0 \leq n + 2^{k-1} \leq 2^k - 1$

Le nombre 0 est codé par 2^{k-1} : $1\,\underbrace{0 \quad 0}_{k-1 \text{ fois}}$

$2^{k-1} - 1$ par $\underbrace{1\ 1 \quad 1}_{k \text{ fois}}$

-2^{k-1} est codé par $\underbrace{0\ 0 \quad 0}_{k \text{ fois}}$

La valeur 2^{k-1} s'appelle le biais ou le décalage

Exposant biaisé

- Si $0 \leq c(n) \leq 2^k - 1$, alors il code l'entier signé $n = c(n) - 2^{k-1}$
- Si $-2^{k-1} \leq n \leq 2^{k-1} - 1$, alors son code est $c(n) = n + 2^{k-1}$
- codage souvent utilisé dans des circuits qui ne peuvent pas traiter des nombres positifs et négatifs. Par exemple en traitement du signal (DSP).
- Pour un codage sur k = 4 bits et un biais de $2^{4-1} = 8$ $-8 \leq n \leq +7$

	CA2	biaisé		CA2	biaisé
0 =	0 0 0 0	1 0 0 0	-8 =	1 0 0 0	0 0 0 0
1 =	0 0 0 1	1 0 0 1	-7 =	1 0 0 1	0 0 0 1
2 =	0 0 1 0	1 0 1 0	-6 =	1 0 1 0	0 0 1 0
3 =	0 0 1 1	1 0 1 1	-5 =	1 0 1 1	0 0 1 1
4 =	0 1 0 0	1 1 0 0	-4 =	1 1 0 0	0 1 0 0
5 =	0 1 0 1	1 1 0 1	-3 =	1 1 0 1	0 1 0 1
6 =	0 1 1 0	1 1 1 0	-2 =	1 1 1 0	0 1 1 0
7=	0 1 1 1	1 1 1 1	-1 =	1 1 1 1	0 1 1 1

Exposant biaisé : exemple de codage

Représentation avec le format

Signe mantisse	Exposant	Mantisse normalisée
1 bit	4 bits	8 bits

exemple $(0{,}015)_8$

$(0{,}015)_8 = (0{,}000001101)_2 = (0{,}1101)_2 \,.\, (2^{-5})_{10}$

Mantisse positive, donc bit de signe égal à 0

Mantisse normalisée $(0,\ 1101)_2$

Exposant $(-5)_{10}$ Codage biaisé en complément à deux sur 4 bits

le biais est $2^{4-1} = 8$, l'exposant biaisé est $-5 + 8 = 3_{10}$

écrit en binaire sur 4 bits, l'exposant est codé par $(0011)_2$

Représentation du nombre : 0 0011 11010000

Intérêt de l'exposant biaisé - comparaison

- Un comparateur logique, opérant bit à bit et de gauche à droite, peut facilement comparer les nombres.
- Il suffit d'aller de gauche à droite, le premier nombre qui a un 1 quand l'autre a un 0 est le plus grand (à part pour le signe).

0 0010 11010000   0 0011 11010100 est plus grand

0 0011 11010000 est plus grand   0 0011 11010000

- Faire des additions en CA2 n'a plus de sens avec des codes biaisés, c'est uniquement pour le codage et la comparaison

Procédé du bit caché (en base 2)

- En base 2, la représentation normalisée de la mantisse commence par 1

  Le bit le plus significatif ne représente aucune information, on peut donc supposer sa présence de façon implicite
- Sur un mot de p bits, on gagne un bit pour avoir une mantisse de p + 1 bits significatifs. On double les mantisses représentables
- En base b > 2

  la mantisse normalisée commence par $d_{-1} \in \{1, 2, \ldots, b - 1\}$

  Le procédé du bit caché ne s'applique pas

Procédé du bit caché

- Exemple avec une Mantisse sur p = 3 bits
- Mantisse normalisée : 4 mantisses distinctes

  100 = $(0,5)_{10}$ 101 = $(0,625)_{10}$ 110 = $(0,75)_{10}$ 111 = $(0,875)_{10}$

- Avec un bit caché qui vaut 1 8 mantisses possibles :

  1 000 soit $(0,5000)_{10}$ 1 100 soit $(0,7500)_{10}$
  1 001 soit $(0,5625)_{10}$ 1 101 soit $(0,8125)_{10}$
  1 010 soit $(0,6250)_{10}$ 1 110 soit $(0,8750)_{10}$
  1 011 soit $(0,6875)_{10}$ 1 111 soit $(0,9375)_{10}$

Bit caché et exposant biaisé : Exemple

Biais : $2^{k-1} = 4$

La Représentation comprend

un bit de signe, un exposant biaisé de 3 bits et une mantisse de 3 bits, avec utilisation du bit caché.

Pour trouver le successeur d'un nombre, il suffit de prendre tout le code et d'ajouter 1 en binaire.

Code	Valeur binaire (notation sc.)	Valeur binaire (notation à virgule)	Valeur décimale
0 110 000	$0,1000.10^{10}$	10,00	2,00
0 110 001	$0,1001.10^{10}$	10,01	2,25
0 110 010	$0,1010.10^{10}$	10,10	2,50
0 110 011	$0,1011.10^{10}$	10,11	2,75
0 110 100	$0,1100.10^{10}$	11,00	3,00
0 110 101	$0,1101.10^{10}$	11,01	3,25
0 110 110	$0,1110.10^{10}$	11,10	3,50
0 110 111	$0,1111.10^{10}$	11,11	3,75
0 111 000	$0,1000.10^{11}$	100,0	4,00

Bilan : bit caché et exposant biaisé

Avantages du codage utilisant un bit caché et un exposant biaisé

- on peut facilement comparer deux nombres
- on obtient le successeur d'un nombre en ajoutant simplement 1 au code
- on augmente la précision de la mantisse sans coût matériel

Format flottant IEEE

- Le standard IEEE utilise le bit caché pour la mantisse : il est de poids 20 et la mantisse est donc 1, d_{-1} d_{-p}
- Le décalage/biais de l'exposant sur k bits est de $2^{k-1} - 1$ et on ne représente que des exposants signés de $-2^{k-1} + 2$ à $2^{k-1} - 1$.

  Pour k = 8, biais égal à 127 et les exposants vont de -126 à +127.

- Les exposants biaisés 0 et $2^k - 1$ sont utilisés pour représenter des nombres sous-normaux, l'infini `Inf` et un nombre non défini `NaN`
- Résumé format IEEE s e d_{-1} d_{-2} d_{-p}

Nom	Taille	Signe s	Exposant e	Biais	Mantisse	Précision p + 1	Chiffres significatifs
float	32 bits	1 bit	8 bits	127	23 bits	24	7
double	64 bits	1 bit	11 bits	1023	52 bits	54	16

Conclusion

- La représentation des réels en machine nécessite de choisir la taille mémoire :

  souvent 4 octets ou 8 octets, parfois 16 octets

- Les nombres réels représentables en machine sont en nombre fini
- Les calculs en flottant peuvent provoquer des underflow, overflow et erreurs d'arrondi

  Certains standards permettent de gérer des exceptions : Inf, NaN, nombres sous-normaux

- On mesure la puissance d'une unité de calcul en virgule flottante en FLOPS (en anglais, FLoating point Operations Per Second).

  En juin 2013, l'ordinateur Tianhe-2 de la NUDT, Chine, a effectué 33,86 petaFLOPS = 33,86 10^{15} FLOPS contre 10^{10} FLOPS pour un processeur "normal" à 2,5 GHz.

Conclusion

- Ne pas faire des tests d'égalité entre des nombres en virgule flottante
- Faire des tests d'inégalité :

  $\boxed{\text{valeur absolue de x < epsilon ?}}$

  où epsilon est la précision, par exemple 10^{-6}

  au lieu de $\boxed{x = 0 ?}$

Propagation d'erreurs

- Opérations entre nombres arrondis, on obtient un résultat dont l'erreur est en général plus importante que les erreurs initiales
- Exemple

Soit x un nombre dont on connaît une valeur approchée x_0 à Δ près

Δ est appelée "incertitude"

On note $x = x_0 \pm \Delta$

$\pi = 3{,}14 \pm \Delta$ avec $\Delta = 0{,}01$ $\pi = 3{,}14159265359\ldots$

$\pi + \pi$ connu à 2Δ près : $\pi + \pi = 6{,}28 \pm 0{,}02$

25

Mathématiques et informatique Sciences Université de Paris

Propagation d'erreurs

- Lors de l'addition/soustraction, on additionne les incertitudes

Si $x = x_0 \pm \Delta$ et $y = y_0 \pm \Delta'$, $x - y = x_0 - y_0 \pm (\Delta + \Delta')$

- Lors de la multiplication, la formule est la suivante :

$$xy = x_0 y_0 \pm (|x_0|\Delta' + |y_0|\Delta)$$

Exemple : $x = 2 \pm 0{,}01$ et $y = -3 \pm 0{,}2$

$xy = -6 \pm \Delta$ avec $\Delta = 2 \,.\, 0{,}2 + 3 \,.\, 0{,}01 = 0{,}43$

26

Mathématiques et informatique Sciences Université de Paris

Propagation d'erreurs

- Exemple : On a $x = 3/4 \pm 0{,}08$ $y = 10 \pm 0{,}1$ et $z = -3 \pm 0{,}01$.
- Calculer l'incertitude sur $x(y + z)$.

$y + z = 7 \pm (0{,}01 + 0{,}1) = 7 \pm 0{,}11$

donc

$x(y + z) = (3/4 \pm 0{,}08)(7 \pm 0{,}11) = 21/4 \pm (7 \,.\, 0{,}08 + 0{,}11 \,.\, 3/4) = 5{,}25 \pm 0{,}6425$

- En effet, si on a le pire cas de figure : $x = 0{,}75 + 0{,}08$; $y = 10{,}1$ et $z = -3 + 0{,}01$

$x(y + z) = 5{,}8266$

27

Mathématiques et informatique Sciences Université de Paris

Addition de grandes sommes

- Soit n = 1 million, $x_1, \ldots, x_n$ des données pour lesquelles on a une incertitude de Δ

$$S = \sum_i^n x_i$$

l'incertitude sur S est de $n\Delta$

$$x = \frac{1}{n}\sum_{i=1}^n x_i$$

l'incertitude sur x est $n.1/n.\Delta = \Delta$

l'incertitude ne croît pas lors du passage à la moyenne

28

Mathématiques et informatique Sciences Université de Paris