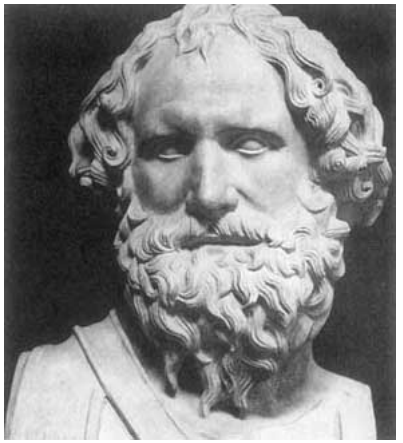
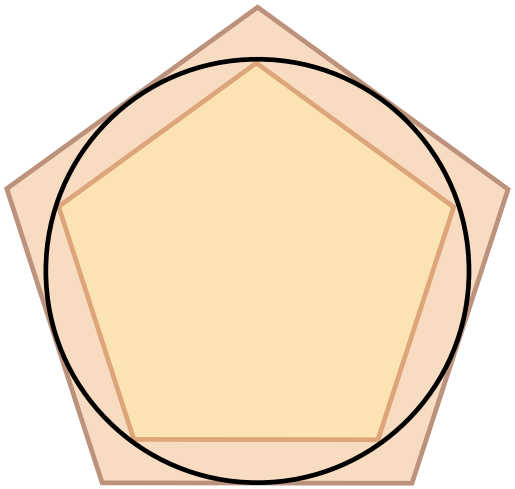
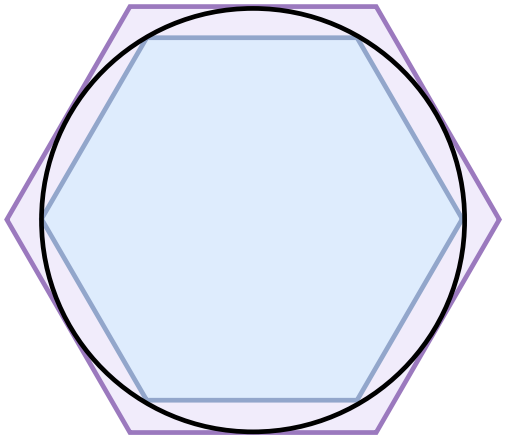
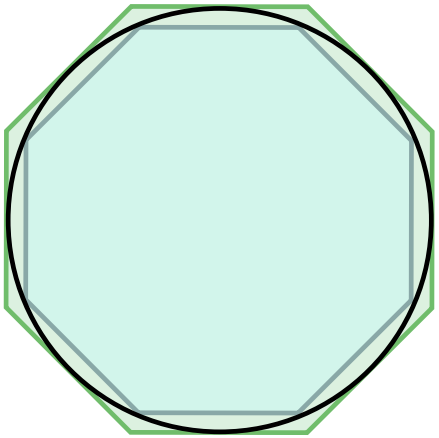
Archimède et un encadrement de π









$$\frac{223}{71} < \pi < \frac{22}{7}$$

Arithmétique par intervalles et bibliothèques deux éclairages

Nathalie Revol
Nathalie.Revol@inria.fr

Groupe de travail Lyon-Grenoble « Vérification »
18 avril 2017

Table des matières

Compter sans se tromper : arithmétique par intervalles

Principe

Nombres remplacés par des intervalles.

π remplacé par $[3.14159, 3.14160]$ ou $[3.14, 3.15]$ ou $[3, 4]$.

Théorème fondamental (tu ne mentiras point) : l'intervalle contient la valeur exacte.

Compter sans se tromper : arithmétique par intervalles

Exemple

Contenu de mon porte-monnaie : entre 10 Euros et 20 Euros,
$\in [10, 20]$ Euros.

Compter sans se tromper : arithmétique par intervalles

Exemple

Contenu de mon porte-monnaie : entre 10 Euros et 20 Euros,
$\in [10, 20]$ Euros.

Contenu de votre porte-monnaie : entre 5 Euros et 10 Euros,
$\in [5, 10]$ Euros.

Compter sans se tromper : arithmétique par intervalles

Exemple

Contenu de mon porte-monnaie : entre 10 Euros et 20 Euros,
$\in [10, 20]$ Euros.

Contenu de votre porte-monnaie : entre 5 Euros et 10 Euros,
$\in [5, 10]$ Euros.

Ensemble, nous avons entre 15 et 30 Euros,
$[10, 20] + [5, 10] = [15, 30]$ Euros.

Arithmétique par intervalles : premier problème

Contenu de mon porte-monnaie : entre 10 Euros et 20 Euros,
$\in [10, 20]$ Euros.

Arithmétique par intervalles : premier problème

Contenu de mon porte-monnaie : entre 10 Euros et 20 Euros,
$\in [10, 20]$ Euros.

Je vais voir vos grands-parents, qui me donnent une enveloppe pour vous, contenant entre 10 et 20 Euros.
Je range l'argent dans mon porte-monnaie, il contient maintenant entre 20 et 40 Euros : $[10, 20] + [10, 20] = [20, 40]$ Euros.

Arithmétique par intervalles : premier problème

Contenu de mon porte-monnaie : entre 10 Euros et 20 Euros,
$\in [10, 20]$ Euros.

Je vais voir vos grands-parents, qui me donnent une enveloppe pour vous, contenant entre 10 et 20 Euros.
Je range l'argent dans mon porte-monnaie, il contient maintenant entre 20 et 40 Euros : $[10, 20] + [10, 20] = [20, 40]$ Euros.

Je vous donne votre argent, entre 10 et 20 Euros.
Mon porte-monnaie contient $[20, 40] - [10, 20] = [0, 30]$ Euros.

Arithmétique par intervalles : premier problème

Contenu de mon porte-monnaie : entre 10 Euros et 20 Euros,
$\in [10, 20]$ Euros.

Je vais voir vos grands-parents, qui me donnent une enveloppe pour vous, contenant entre 10 et 20 Euros.
Je range l'argent dans mon porte-monnaie, il contient maintenant entre 20 et 40 Euros : $[10, 20] + [10, 20] = [20, 40]$ Euros.

Je vous donne votre argent, entre 10 et 20 Euros.
Mon porte-monnaie contient $[20, 40] - [10, 20] = [0, 30]$ Euros.

Autrement dit,

$$\text{porte-monnaie} + \text{enveloppe} - \text{enveloppe} \neq \text{porte-monnaie}.$$

Arithmétique par intervalles : deuxième problème

Rappel : un nombre au carré est toujours positif $\Rightarrow$ on ne peut définir la racine carrée d'un nombre que s'il est positif.

Arithmétique par intervalles : deuxième problème

Rappel : un nombre au carré est toujours positif $\Rightarrow$ on ne peut définir la racine carrée d'un nombre que s'il est positif.

Comment définir $\sqrt{[-1,2]}$?

Arithmétique par intervalles : deuxième problème

Rappel : un nombre au carré est toujours positif $\Rightarrow$ on ne peut définir la racine carrée d'un nombre que s'il est positif.

Comment définir $\sqrt{[-1,2]}$?

Par convention, $\sqrt{[-1,2]} = \sqrt{[0,2]} = [0,\sqrt{2}]$.
Il faut signaler qu'il y a eu un problème :

Arithmétique par intervalles : deuxième problème

Rappel : un nombre au carré est toujours positif $\Rightarrow$ on ne peut définir la racine carrée d'un nombre que s'il est positif.

Comment définir $\sqrt{[-1,2]}$?

Par convention, $\sqrt{[-1,2]} = \sqrt{[0,2]} = [0,\sqrt{2}]$.
Il faut signaler qu'il y a eu un problème :

- comment ? en « décorant » les intervalles 

Arithmétique par intervalles : deuxième problème

Rappel : un nombre au carré est toujours positif $\Rightarrow$ on ne peut définir la racine carrée d'un nombre que s'il est positif.

Comment définir $\sqrt{[-1,2]}$?

Par convention, $\sqrt{[-1,2]} = \sqrt{[0,2]} = [0,\sqrt{2}]$.
Il faut signaler qu'il y a eu un problème :

- comment ? en « décorant » les intervalles 
- comment faire pour ne pas pénaliser les performances des calculs sur ordinateur : temps de calcul, utilisation de la mémoire ?

Arithmétique par intervalles : troisième problème

Pour une opération entre des intervalles, il faut effectuer 2 (ou 4 ou...) opérations « habituelles ».

Pour des raisons liées à l'architecture des ordinateurs, il faut 100 fois plus de temps (ou pire) pour exécuter une série de calculs sur des intervalles qu'avec des nombres « habituels ».

Comment faire pour que les calculs soient moins lents ?
S'ils sont 10 à 15 fois moins lents, on est déjà satisfait.

Et la complexité des algorithmes ?
Les problèmes sont presque tous NP-durs.

Arithmétique par intervalles : premier problème

Contenu de mon porte-monnaie : entre 10 Euros et 20 Euros,
$\in [10, 20]$ Euros.

Je vais voir vos grands-parents, qui me donnent une enveloppe pour vous, contenant entre 10 et 20 Euros.
Je range l'argent dans mon porte-monnaie, il contient maintenant entre 20 et 40 Euros : $[10, 20] + [10, 20] = [20, 40]$ Euros.

Je vous donne votre argent, entre 10 et 20 Euros.
Mon porte-monnaie contient $[20, 40] - [10, 20] = [0, 30]$ Euros.

Autrement dit,

$$\text{porte-monnaie} + \text{enveloppe} - \text{enveloppe} \neq \text{porte-monnaie}.$$

Arithmétique par intervalles : premier problème

Solution ?

Écrire les formules utilisées dans les algorithmes différemment, avec soin.

Arithmétique par intervalles : premier problème

Solution ?

Écrire les formules utilisées dans les algorithmes différemment, avec soin.

On ne considérera pas cette question ici.

Arithmétique par intervalles : deuxième problème

Rappel : un nombre au carré est toujours positif $\Rightarrow$ on ne peut définir la racine carrée d'un nombre que s'il est positif.

Comment définir $\sqrt{[-1,2]}$?

Par convention, $\sqrt{[-1,2]} = \sqrt{[0,2]} = [0,\sqrt{2}]$.
Il faut signaler qu'il y a eu un problème :

- comment ? en « décorant » les intervalles 
- comment faire pour ne pas pénaliser les performances des calculs sur ordinateur : temps de calcul, utilisation de la mémoire ?

Arithmétique par intervalles : deuxième problème

Solution ?

Choisir une convention et s'y tenir.

Arithmétique par intervalles : deuxième problème

Solution ?

Choisir une convention et s'y tenir.

Norme IEEE 1788

C'est le premier point de vue adopté :

- quels sont les éléments essentiels de cette norme ?
- quelles sont les bibliothèques qui respectent cette norme ?

Arithmétique par intervalles : troisième problème

Pour une opération entre des intervalles, il faut effectuer 2 (ou 4 ou . . .) opérations « habituelles ».

Pour des raisons liées à l'architecture des ordinateurs, il faut 100 fois plus de temps (ou pire) pour exécuter une série de calculs sur des intervalles qu'avec des nombres « habituels ».

Comment faire pour que les calculs soient moins lents ?
S'ils sont 10 à 15 fois moins lents, on est déjà satisfait.

Et la complexité des algorithmes ?
Les problèmes sont presque tous NP-durs.

Arithmétique par intervalles : troisième problème

Solutions ?

? ? ?

Arithmétique par intervalles : troisième problème

Solutions ?
? ? ?

Quelques pistes :

- précision des calculs
- représentation

Précision des calculs : c'est le deuxième point de vue abordé ici.

Sélection de quelques bibliothèques

- **General-purpose libraries :** `IntLib` utilisé par `GlobSol`, `C-XSC`, `fi_lib`, `Octave interval`, `Moore`, `Arb`, `Mathemagix`, `libieee1788`, `MPFI`, `unum` ;

Sélection de quelques bibliothèques

- **General-purpose libraries :** `IntLib` utilisé par `GlobSol`, `C-XSC`, `fi_lib`, `Octave interval`, `Moore`, `Arb`, `Mathemagix`, `libieee1788`, `MPFI`, `unum` ;
- **Bibliothèques spécialisées pour l'algèbre linéaire :** `C-XSC`, `IntLab`, `Profil-BIAS` ;

Sélection de quelques bibliothèques

- **General-purpose libraries :** `IntLib` utilisé par `GlobSol`, `C-XSC`, `fi_lib`, `Octave interval`, `Moore`, `Arb`, `Mathemagix`, `libieee1788`, `MPFI`, `unum` ;
- **Bibliothèques spécialisées pour l'algèbre linéaire :** `C-XSC`, `IntLab`, `Profil-BIAS` ;
- **Bibliothèques spécialisées pour l'intégration des EDO :** `AWA`, `CAPD`, `COSY`, `VSPODE`, (`VNODE` : `fi_lib` `Profil-BIAS`) ;

Sélection de quelques bibliothèques

- **General-purpose libraries :** `IntLib` utilisé par `GlobSol`, `C-XSC`, `fi_lib`, `Octave interval`, `Moore`, `Arb`, `Mathemagix`, `libieee1788`, `MPFI`, `unum` ;
- **Bibliothèques spécialisées pour l'algèbre linéaire :** `C-XSC`, `IntLab`, `Profil-BIAS` ;
- **Bibliothèques spécialisées pour l'intégration des EDO :** `AWA`, `CAPD`, `COSY`, `VSPODE`, (`VNODE` : `fi_lib` `Profil-BIAS`) ;
- **Bibliothèques pour la résolution de contraintes :** `Gaol`, `IBEX-Quimper`, `Realpaver`, (`Alias` : utilisant `Profil-BIAS`) ;

Sélection de quelques bibliothèques

- **General-purpose libraries :** `IntLib` utilisé par `GlobSol`, `C-XSC`, `fi_lib`, `Octave interval`, `Moore`, `Arb`, `Mathemagix`, `libieee1788`, `MPFI`, `unum` ;
- **Bibliothèques spécialisées pour l'algèbre linéaire :** `C-XSC`, `IntLab`, `Profil-BIAS` ;
- **Bibliothèques spécialisées pour l'intégration des EDO :** `AWA`, `CAPD`, `COSY`, `VSPODE`, (`VNODE` : `fi_lib` `Profil-BIAS`) ;
- **Bibliothèques pour la résolution de contraintes :** `Gaol`, `IBEX-Quimper`, `Realpaver`, (`Alias` : utilisant `Profil-BIAS`) ;
- **Bibliothèques pour l'optimisation, avec ou sans contraintes :** `COSY-GO`, `GlobSol`, (`Baron` : `NEOS`, `Coconut` : construit sur `fi_lib` et `Jail`, `GloptLab` : sur `IntLab`) ;

Sélection de quelques bibliothèques

- **General-purpose libraries :** IntLib utilisé par GlobSol, C-XSC, fi_lib, Octave interval, Moore, Arb, Mathemagix, libieee1788, MPFI, unum ;
- **Bibliothèques spécialisées pour l'algèbre linéaire :** C-XSC, IntLab, Profil-BIAS ;
- **Bibliothèques spécialisées pour l'intégration des EDO :** AWA, CAPD, COSY, VSPODE, (VNODE : fi_lib Profil-BIAS) ;
- **Bibliothèques pour la résolution de contraintes :** Gaol, IBEX-Quimper, Realpaver, (Alias : utilisant Profil-BIAS) ;
- **Bibliothèques pour l'optimisation, avec ou sans contraintes :** COSY-GO, GlobSol, (Baron : NEOS, Coconut : construit sur fi_lib et Jail, GloptLab : sur IntLab) ;
- **Bibliothèques implantant des variantes de l'AI :** modèles de Taylor dans COSY et C-XSC, arithmétique affine dans Fluctuat et Ibex.

	intervals & arith. op.	exact dot product	reverse op. (+ and -)	exceptions handling	variants of int. arith.
applications	all	linear algebra	constraints	Newton constraints	ODE verif.
C-XSC	✓	✓		?	Taylor
fi_lib	✓			?	
IntLib	✓			kind of	Taylor
Octave	✓	✓	✓	✓	
Arb	✓				
Mathemagix	✓				
libieee1788	✓	✓	✓	✓	
MPFI	✓			✓	
Profil-BIAS	?	✓			
IntLab	✓	✓			
AWA	?	kind of	kind of		Taylor
CAPD		kind of		✓	
COSY					Taylor
VSPODE	?				Taylor
Ibex	✓		✓	✓	
COSY-GO	NA				Taylor
GlobSol	cset or ✓ ?			kind of	Taylor
Fluctuat	?		planned		

Table des matières

IEEE-1788 standard : the big picture

LEVEL1 mathematics	
LEVEL2 implementation or discretization	
LEVEL3 computer representation	
LEVEL4 bits	

IEEE-1788 standard : the big picture

LEVEL1 mathematics	**objects** representation (no mid–rad...) constructors
LEVEL2 implementation or discretization	
LEVEL3 computer representation	
LEVEL4 bits	

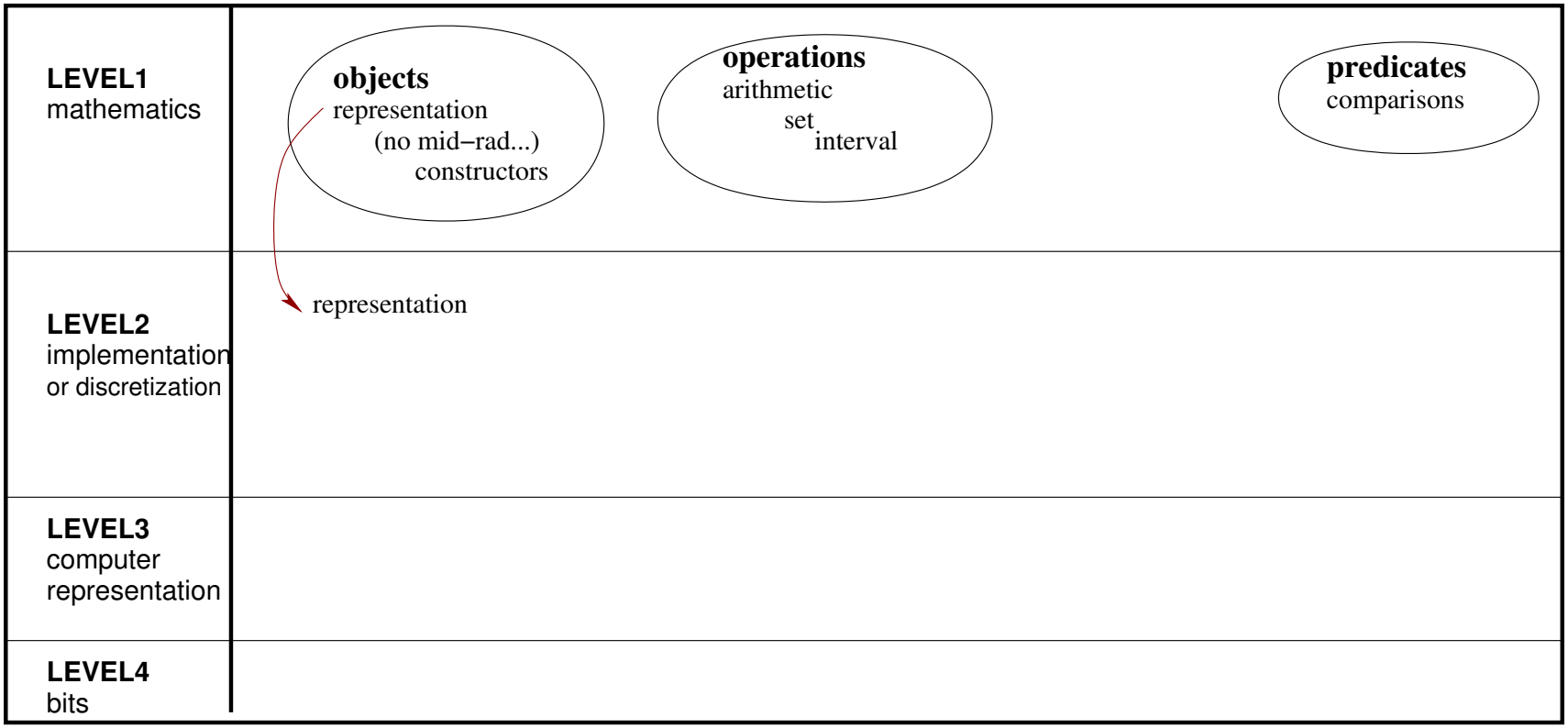
IEEE-1788 standard : the big picture

LEVEL1 mathematics	**objects** representation (no mid–rad...) constructors **operations** arithmetic set interval
LEVEL2 implementation or discretization	
LEVEL3 computer representation	
LEVEL4 bits	

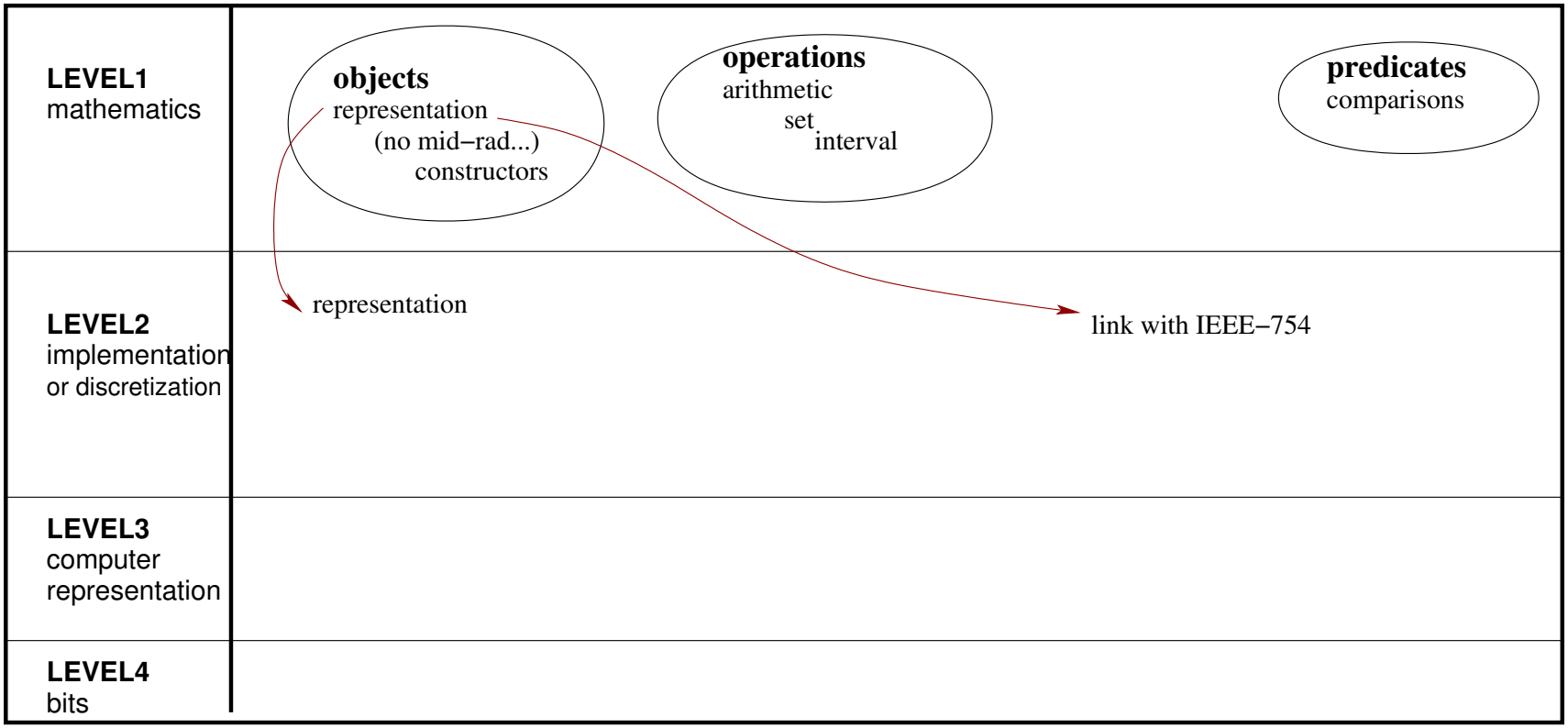
IEEE-1788 standard : the big picture

LEVEL1 mathematics	**objects** representation (no mid–rad...) constructors **operations** arithmetic set interval **predicates** comparisons
LEVEL2 implementation or discretization	
LEVEL3 computer representation	
LEVEL4 bits	

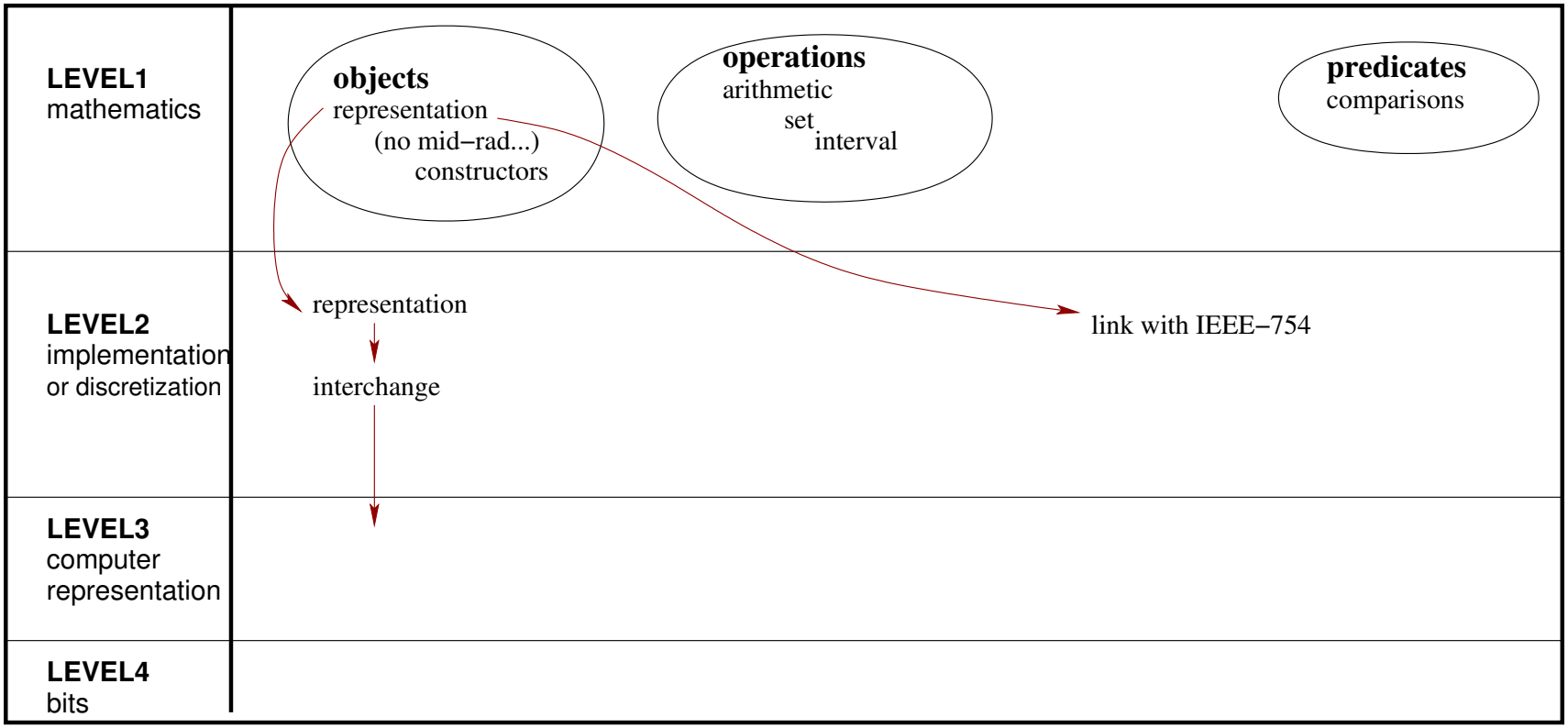
IEEE-1788 standard : the big picture



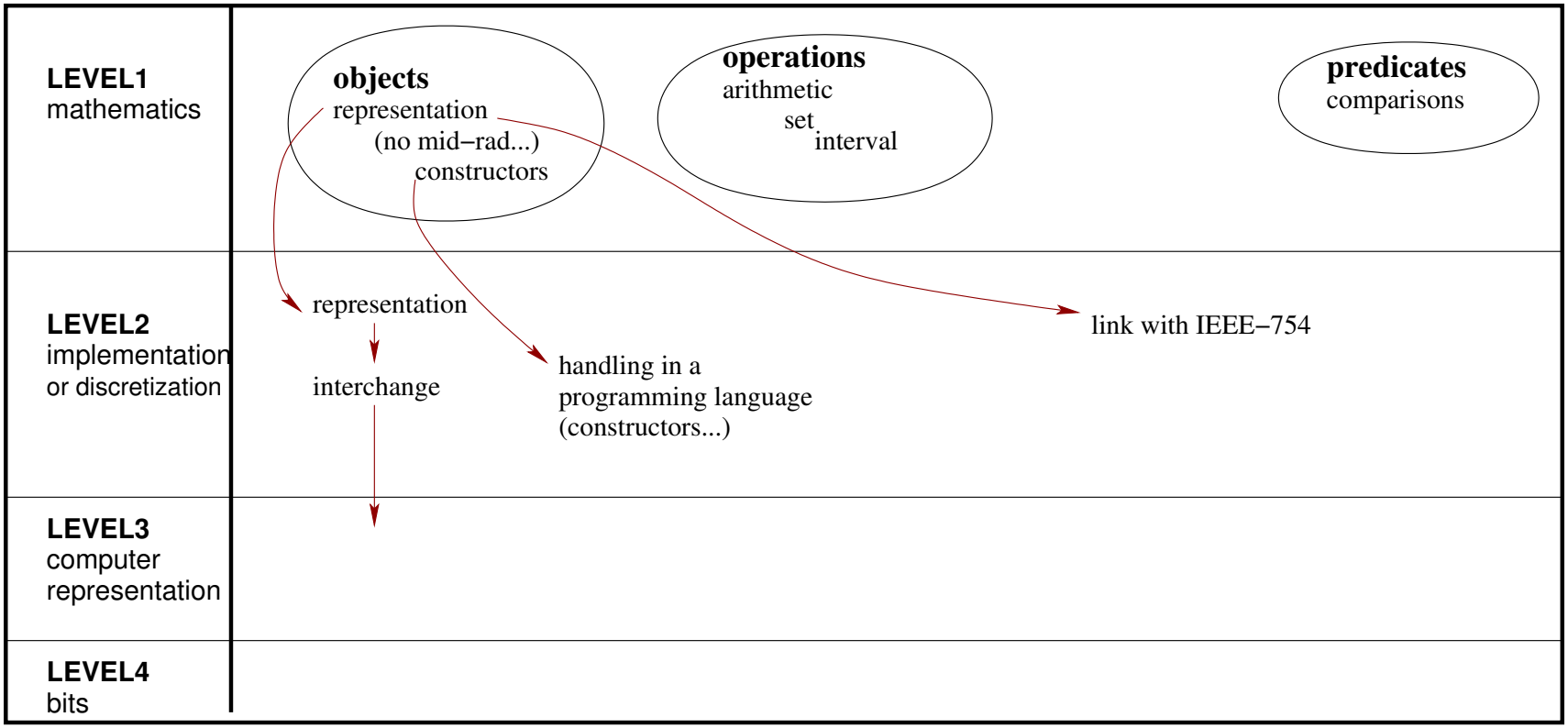
IEEE-1788 standard : the big picture



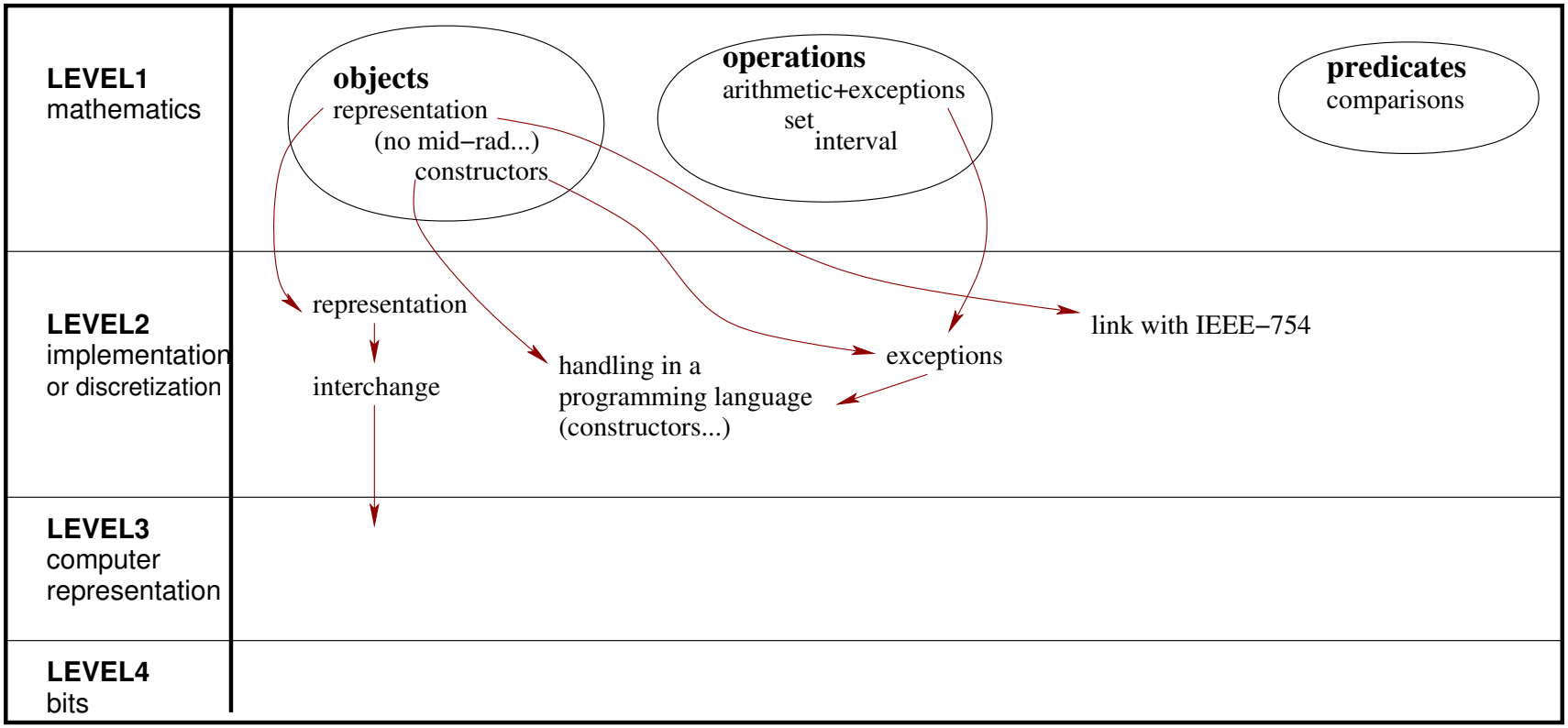
IEEE-1788 standard : the big picture



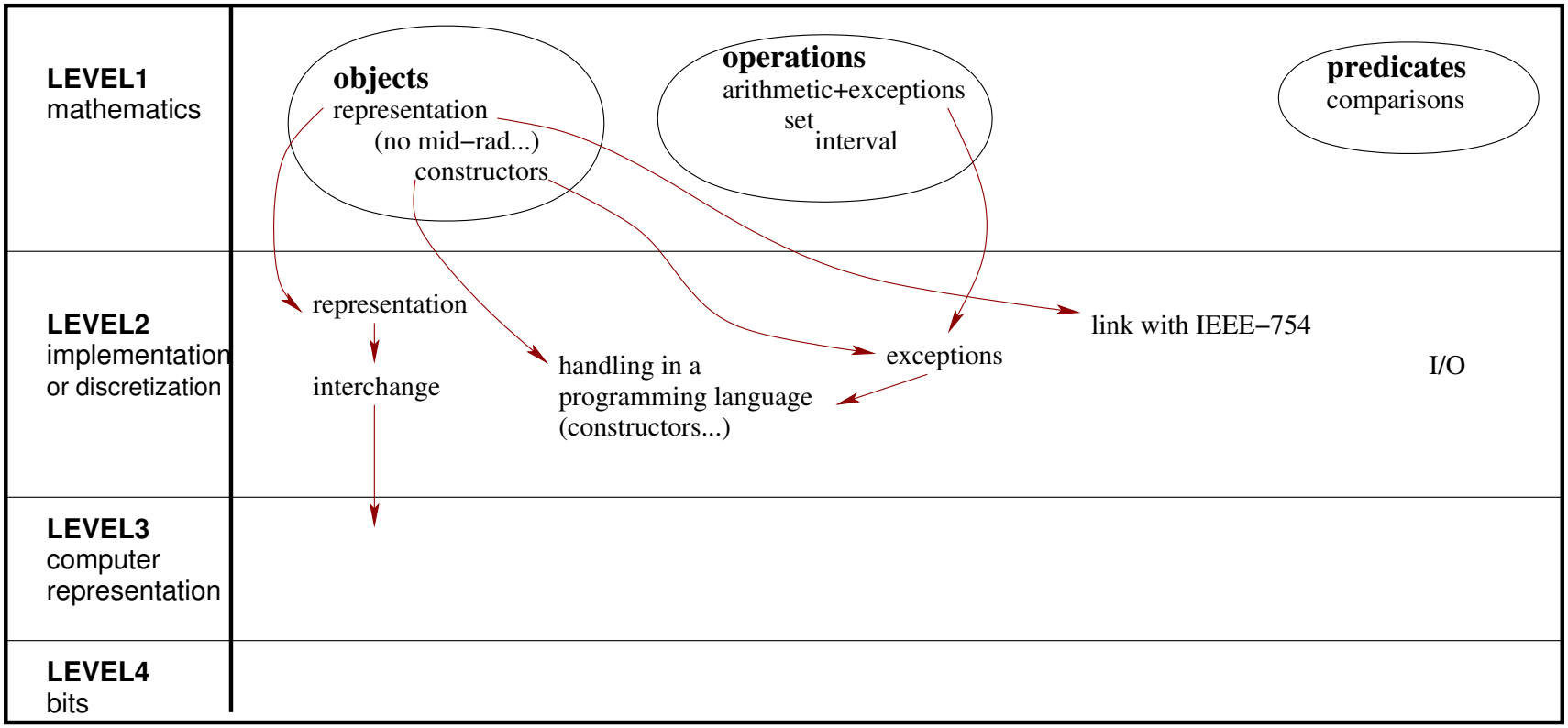
IEEE-1788 standard : the big picture



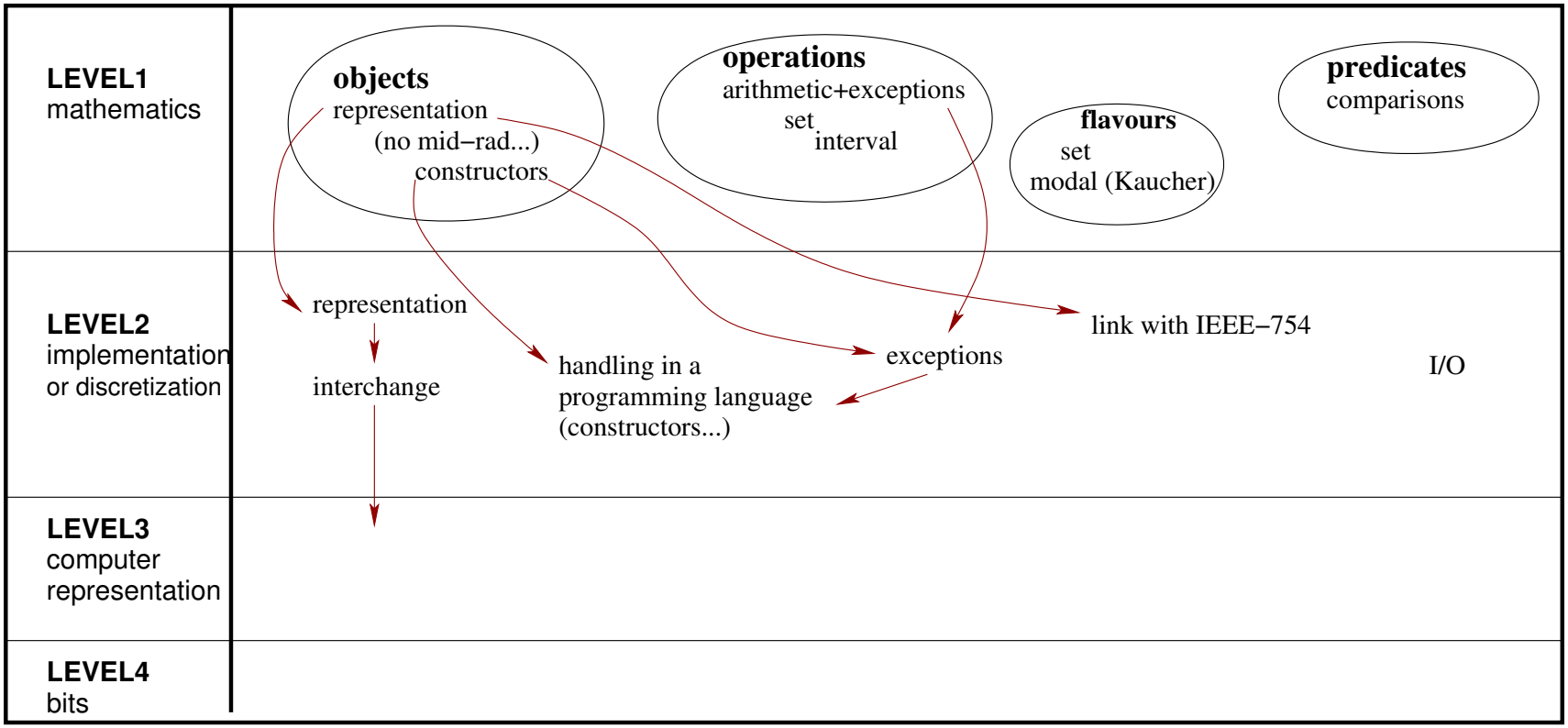
IEEE-1788 standard : the big picture



IEEE-1788 standard : the big picture



IEEE-1788 standard : the big picture



IEEE-1788 standard : the big picture

Opération recommandée : produit scalaire exact ou **edp** pour **exact dot product**.
Ce n'est pas une opération portant sur les intervalles.

Bibliothèques implantant la norme IEEE-1788

Celles qui ont été développées après la mise au point de la norme, en guise de validation de la norme :

- Octave interval
- libieee1788
- jInterval : interface Java de différentes bibliothèques + tests de conformité IEEE 1788

Bibliothèques implantant la norme IEEE-1788

Celles qui ont été développées après la mise au point de la norme, en guise de validation de la norme :

- **Octave interval**
- **libieee1788**
- jInterval : interface Java de différentes bibliothèques + tests de conformité IEEE 1788

On s'intéressera à Octave interval.

DEMO

Table des matières

Influence of the computing precision (1/2)

Influence on one value :

$$t \notin \mathbb{F} \quad \Rightarrow \quad t \in \Big[\mathrm{RD}(t), \mathrm{RU}(t)\Big] \quad \Rightarrow \quad \mathrm{RU}(t) - \mathrm{RD}(t) \leq 2u|t|.$$

Influence on one interval operation : the overestimation of the result is proportional to 4 ulp :
$\mathrm{w}(\widehat{\boldsymbol{x} \operatorname{op} \boldsymbol{y}}) - \mathrm{w}(\boldsymbol{x} \operatorname{op} \boldsymbol{y}) \leq 4u|\boldsymbol{x} \operatorname{op} \boldsymbol{y}|)$.

Influence on an interval computation : theoretically, the overestimation of the result is proportional to the ulp :
$\mathrm{w}(\hat{\boldsymbol{x}}) - \mathrm{w}(\boldsymbol{x}) = \mathcal{O}(2^{-p}|\boldsymbol{x}|)$ where p is the computing precision.

Influence of the computing precision (2/2)

Influence on an interval computation : in practice,

- use the midpoint-radius representation for thin intervals : the radius accounts for roundoff errors,
- use iterative refinement to reduce the width,
- use higher precision for critical intermediate computations (residual) to hide the effect of the computing precision,

and get $\mathrm{w}(\hat{\boldsymbol{x}}) - \mathrm{w}(\boldsymbol{x}) \simeq 2^{-p}|\boldsymbol{x}|$, i.e. the best possible result.

Examples : linear systems solving, Newton iteration.

Higher precision : extended / arbitrary

Extended precision (double-double, triple-double) : (Moler, Priest, Dekker, Knuth, Shewchuk, Bailey...)
a number is represented as the sum of 2 (or 3 or ...) floating-point numbers. Do not evaluate the sum using floating-point arithmetic ! Double-double arith. is implemented using IEEE-754 FP arith.

Arbitrary precision : the precision is chosen by the user, the only limit being the computer's memory.
Arithmetic is implemented in software, e.g. MPFR (**Zimmermann et al.**), MPFI (**Revol, Rouillier et al.**), (**Yamamoto, Krämer et al.**).

Tradeoff between accuracy and efficiency (and memory) :
double-double : accuracy "$\times 2$", $\leq$ 1 order of magnitude slower
arbitrary prec. : accuracy "∞", $\geq$ 1-2 order of magnitude slower (provided Higham's rule of thumb applies).

Higher precision : extended / arbitrary

Higher precision : extended / arbitrary

Extended precision (double-double, triple-double) : (Moler, Priest, Dekker, Knuth, Shewchuk, Bailey...)
a number is represented as the sum of 2 (or 3 or ...) floating-point numbers. Do not evaluate the sum using floating-point arithmetic! Double-double arith. is implemented using IEEE-754 FP arith.

Arbitrary precision : the precision is chosen by the user, the only limit being the computer's memory.
Arithmetic is implemented in software, e.g. MPFR (**Zimmermann et al.**), MPFI **(Revol, Rouillier et al.), (Yamamoto, Krämer et al.**).

Tradeoff between accuracy and efficiency (and memory) :
double-double : accuracy "$\times 2$", $\leq$ 1 order of magnitude slower
arbitrary prec. : accuracy "∞", $\geq$ 1-2 order of magnitude slower (provided Higham's rule of thumb applies).

À dire : dire qu'il existe d'autres approches destinées à améliorer la précision du résultat, soit en jouant sur la précision des calculs – mais en restant limitée – soit en luttant contre la décorrélation des variables.□

Bibliothèques offrant une précision étendue

- Arb
- Mathemagix
- libieee1788
- MPFI

Bibliothèques offrant une précision étendue

- Arb
- Mathemagix
- libieee1788
- MPFI

On s'intéressera à MPFI.

Multiple Precision Interval Arithmetic

What is MPFI ?

- based on MPFR library : arbitrary precision :
- MPFR stands for *Multiple Precision Reliable Floating-point library* :
- MPFI stands for *Multiple Precision reliable Floating-point Interval library* :
- the computing precision of each operation can be specified :
- no limit apart from the memory of your computer.

DEMO

Arithmétique par intervalles et bibliothèques
Précision
Bibliothèques offrant une précision arbitraire
Multiple Precision Interval Arithmetic

Multiple Precision Interval Arithmetic

What is MPFI ?

- based on MPFR library : arbitrary precision :
- MPFR stands for *Multiple Precision Reliable Floating-point library* :
- MPFI stands for *Multiple Precision reliable Floating-point Interval library* :
- the computing precision of each operation can be specified :
- no limit apart from the memory of your computer.

DEMO

À dire : MPFI représente les intervalles par leurs extrémités, et avec une précision arbitraire. Pour cela on a besoin d'une bibliothèque d'arithmétique en précision arbitraire sur les flottants : MPFR. Expliquer que le nom MPFR existant, le choix du nom MPFI s'imposait ensuite. Dire qu'il s'agit d'une bibliothèque en C – oui, désolée, pas de version C++ avec opérateurs surchargés. □

Mid-rad representation of intervals

Use of mid-rad representation : better suited for this purpose, as the midpoint corresponds to the floating-point value and the radius accounts for roundoff errors.

with usual precision (floating-point arithmetic available on the processor), cf. IntLab library :
efficient, often does the job.

with arbitrary precision (cf. ARB or Mathemagix library) :
for the midpoint and much less precision for the radius.

Mid-rad representations of intervals : operations

Addition :

$$\begin{aligned} z_m &= \mathrm{RN}(x_m + y_m) \text{ and} \\ z_r &= \mathrm{RU}(x_r + y_r + u \cdot |z_m|). \end{aligned}$$

Multiplication :

$$\begin{aligned} z_m &= \mathrm{RN}(x_m \cdot y_m) \text{ and} \\ z_r &= \mathrm{RU}((|x_m| + x_r) \cdot y_r + x_r \cdot |y_m| + u \cdot |z_m|). \end{aligned}$$

Mid-rad representations of intervals : operations in practice

To avoid costly changes of the rounding modes, use RN and inflate the radii.

Addition :

$$\begin{aligned} z_m &= \mathrm{RN}(x_m + y_m) \text{ and} \\ z_r &= \mathrm{RN}((1+4u)\cdot(x_r + y_r + u\cdot|z_m|)). \end{aligned}$$

Multiplication :

$$\begin{aligned} z_m &= \mathrm{RN}(x_m \cdot y_m) \text{ and} \\ z_r &= \mathrm{RN}((1+4u)\cdot((|x_m| + x_r)\cdot y_r + x_r\cdot|y_m| + u\cdot|z_m|)). \end{aligned}$$

Affine arithmetic Comba, Stolfi and Figueiredo – Fluctuat

Definition : each input or computed quantity x is represented by
$x = x_0 + \alpha_1\varepsilon_1 + \alpha_2\varepsilon_2 + \cdots + \alpha_n\varepsilon_n$
where x_0, α_1, $\ldots \alpha_n$ are known real / floating-point numbers,
and ε_1, ε_2 $\ldots \varepsilon_n$ are symbolic variables $\in [-1, +1]$.
Example : $x \in [3, 7]$ is represented by $x = 5 + 2\varepsilon$.

Operations :
$(x + \sum_k \alpha_k\varepsilon_k) + (y + \sum_k \beta_k\varepsilon_k) = (x + y) + \sum_k(\alpha_k + \beta_k)\varepsilon_k$.
$(x+\sum_k \alpha_k\varepsilon_k)\times(y+\sum_k \beta_k\varepsilon_k) = (x\times y)+\sum_k(x\beta_k+y\alpha_k)\varepsilon_k+\gamma_I\varepsilon_I$
with ε_I a new variable.

Roundoff errors : compute δ_I an upper bound of all roundoff errors and add it to γ_I.

Taylor models, polynomial models

Berz, Hoefkens and Makino 1998, Nedialkov, Neher

Principle : represent a function $f(x)$ for $x \in [-1, 1]$ by a polynomial part $p(x)$ and a remainder part (a big bin) $\boldsymbol{I}$ such that $\forall x \in [-1, 1],\ f(x) \in p(x) + \boldsymbol{I}$.

Operations :

- affine operations : straigthforward ;
- non-affine operations : enclose the nonlinear terms and add this enclosure to the remainder.

Roundoff errors : determine an upper bound b on the roundoff errors and add $[-b, b]$ to the remainder.

Table des matières

Référence

W. Kahan : *How Futile is Mindless Assessment of Roundoff in Floating-Point Computation ?*, 2006.

Référence

W. Kahan : *How Futile is Mindless Assessment of Roundoff in Floating-Point Computation ?*, 2006.

À dire : deuxième partie sur l'utilisation de l'arithmétique par intervalles pour estimer la précision numérique d'un code.
Historiquement, Moore dans les années 60 a créé l'arithmétique par intervalles pour étudier l'impact des erreurs d'arrondi dans ses codes : un intervalle fin en sortie signifie qu'il n'y a pas de problème avec les erreurs d'arrondi, un intervalle large peut signifier qu'il y a un problème, ou alors que l'arithmétique par intervalles a fourni une trop grosse surestimation.
Cette partie est basée sur un article de Kahan, qui tire à boulets rouges sur l'utilisation sans réfléchir, écervelée d'un certain nombre de techniques. Il donne des exemples qui les font échouer, pour montrer qu'appliquer ces techniques sans réfléchir est futile. À nouveau. insistons sur le fait qu'on ne peut pas se dispenser d'être intelligent.
L'autre référence est une introduction à l'arithmétique par intervalles, un court bouquin bien fait qui mentionne aussi l'arithmétique flottante au début.□

Five approaches detailed in Kahan's paper

1. Repeat the computation in arithmetics of increasing precision, increase it until as many as desired of the results' digits agree.
2. Repeat the computation in arithmetic of the same precision but rounded differently, say *Down*, and then *Up*, and maybe *Towards Zero* too, besides *To Nearest*, and compare three or four results.
3. Repeat the computation a few times in arithmetic of the same precision rounding operations randomly, some *Up*, some *Down*, and treat results statistically.
4. Repeat the computation a few times in arithmetic of the same precision but with slightly different input data each time, and see how widely results spread.
5. Perform the computation in *Significance Arithmetic*, or in *Interval Arithmetic*.

The mindless use of these approaches is qualified as "futile" by Kahan.

Five approaches detailed in Kahan's paper

Five approaches detailed in Kahan's paper

1. Repeat the computation in arithmetics of increasing precision, increase it until as many as desired of the results' digits agree.
2. Repeat the computation in arithmetic of the same precision but rounded differently, say *Down*, and then *Up*, and maybe *Towards Zero* too, besides *To Nearest*, and compare three or four results.
3. Repeat the computation a few times in arithmetic of the same precision rounding operations randomly, some *Up*, some *Down*, and treat results statistically.
4. Repeat the computation a few times in arithmetic of the same precision but with slightly different input data each time, and see how widely results spread.
5. Perform the computation in *Significance Arithmetic*, or in *Interval Arithmetic*.

The mindless use of these approaches is qualified as "futile" by Kahan.

À dire : 1) c'est tellement coûteux que pour quasiment le même prix on peut utiliser de l'arithmétique par intervalles en précision arbitraire qui elle au moins donne un encadrement, càd une garantie.□
À dire : 2) cela donne trop peu de résultats possibles, on ne peut pas nécessairement en tirer une conclusion.□
À dire : 3) on peut traiter les résultats statistiquement si on sait ce que l'on fait statistiquement. Dans Cadna, la partie stat est solide.□
À dire : 4) c'est difficile parce qu'il faut savoir de combien faire varier les entrées, et parce qu'il faut savoir interpréter les variations sur les résultats : sont-elles normales ou trop importantes ?
Mais c'est la seule solution quand on n'a pas accès au code, quand on ne peut pas recompiler, quand on ne peut pas modifier le mode d'arrondi.□
À dire : 5) on peut aussi utiliser l'arithmétique par intervalles, ou la Significance Arithmetic qui donne une borne inf – et souvent pessimiste, encore plus pessimiste que l'arithmétique par intervalles – sur le nombre de chiffres corrects du résultat.□

Multiple Precision Interval Arithmetic

Almost foolproof is extendable-precision Interval Arithmetic.
Let's be almost foolproof : let's use MPFI !