Introduction à l'arithmétique par intervalles

Nathalie Revol et Philippe Théveny
INRIA et ENS de Lyon
LIP, ENS de Lyon
Université de Lyon

Ecole *Précision et Reproductibilité en Calcul Numérique*
Fréjus, jeudi 28 mars 2013

Agenda

A brief introduction

Interval arithmetic: replace numbers by intervals and compute.

Fundamental theorem of interval arithmetic:
(or "Thou shalt not lie"):
the exact result (number or set) is contained in the computed interval.

No result is lost, the computed interval is guaranteed to contain every possible result.

A brief introduction

Interval arithmetic: replace numbers by intervals and compute.
Initially: introduced to take into account roundoff errors (Moore 1966)
and also uncertainties (on the physical data...).
Later: computations "in the large", computations with sets.

Interval analysis: develop algorithms for **reliable (or verified, or guaranteed, or certified) computing**,
that are suited for interval arithmetic,
i.e. different from the algorithms from classical numerical analysis.

A brief introduction: examples of applications

- control the roundoff errors, cf. computational geometry
- solve several problems with verified solutions: linear and nonlinear systems of equations and inequalities, constraints satisfaction, (non/convex, un/constrained) global optimization, integrate ODEs e.g. particules trajectories...
- mathematical proofs: cf. Hales' proof of Kepler's conjecture or Tucker's proof that Lorenz system has a strange attractor.

Cf. http://www.cs.utep.edu/interval-comp/

Historical remarks

Who invented Interval Arithmetic?

- **1962:** Ramon Moore defines IA in his PhD thesis and then a rather exhaustive study of IA in a book in 1966

Historical remarks

Who invented Interval Arithmetic?

- **1962:** Ramon Moore defines IA in his PhD thesis and then a rather exhaustive study of IA in a book in 1966
- **1958:** Tsunaga, in his MSc thesis in Japanese

Historical remarks

Who invented Interval Arithmetic?

- **1962:** Ramon Moore defines IA in his PhD thesis and then a rather exhaustive study of IA in a book in 1966
- **1958:** Tsunaga, in his MSc thesis in Japanese
- **1956:** Warmus

Historical remarks

Who invented Interval Arithmetic?

- **1962:** Ramon Moore defines IA in his PhD thesis and then a rather exhaustive study of IA in a book in 1966
- **1958:** Tsunaga, in his MSc thesis in Japanese
- **1956:** Warmus
- **1951:** Dwyer, in the specific case of closed intervals

Historical remarks

Who invented Interval Arithmetic?

- **1962:** Ramon Moore defines IA in his PhD thesis and then a rather exhaustive study of IA in a book in 1966
- **1958:** Tsunaga, in his MSc thesis in Japanese
- **1956:** Warmus
- **1951:** Dwyer, in the specific case of closed intervals
- **1931:** Rosalind Cecil Young in her PhD thesis in Cambridge (UK) has used some formulas

Historical remarks

Who invented Interval Arithmetic?

- **1962:** Ramon Moore defines IA in his PhD thesis and then a rather exhaustive study of IA in a book in 1966
- **1958:** Tsunaga, in his MSc thesis in Japanese
- **1956:** Warmus
- **1951:** Dwyer, in the specific case of closed intervals
- **1931:** Rosalind Cecil Young in her PhD thesis in Cambridge (UK) has used some formulas
- **1927:** Bradis, for positive quantities, in Russian

Historical remarks

Who invented Interval Arithmetic?

- **1962:** Ramon Moore defines IA in his PhD thesis and then a rather exhaustive study of IA in a book in 1966
- **1958:** Tsunaga, in his MSc thesis in Japanese
- **1956:** Warmus
- **1951:** Dwyer, in the specific case of closed intervals
- **1931:** Rosalind Cecil Young in her PhD thesis in Cambridge (UK) has used some formulas
- **1927:** Bradis, for positive quantities, in Russian
- **1908:** Young, for some bounded functions, in Italian

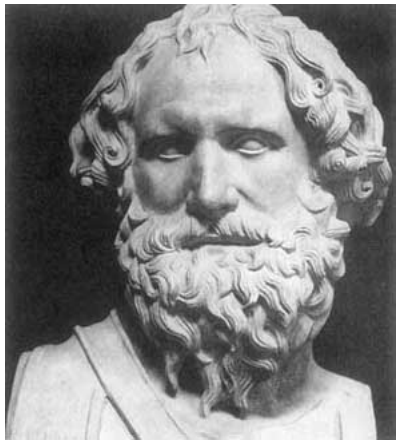
Historical remarks

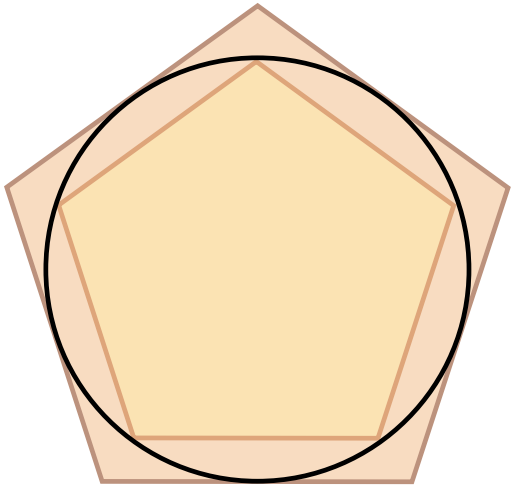
Who invented Interval Arithmetic?

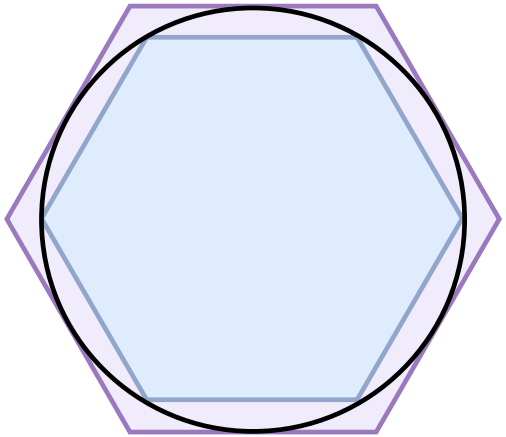
- **1962:** Ramon Moore defines IA in his PhD thesis and then a rather exhaustive study of IA in a book in 1966
- **1958:** Tsunaga, in his MSc thesis in Japanese
- **1956:** Warmus
- **1951:** Dwyer, in the specific case of closed intervals
- **1931:** Rosalind Cecil Young in her PhD thesis in Cambridge (UK) has used some formulas
- **1927:** Bradis, for positive quantities, in Russian
- **1908:** Young, for some bounded functions, in Italian
- **3rd century BC:** Archimedes, to compute an enclosure of π!

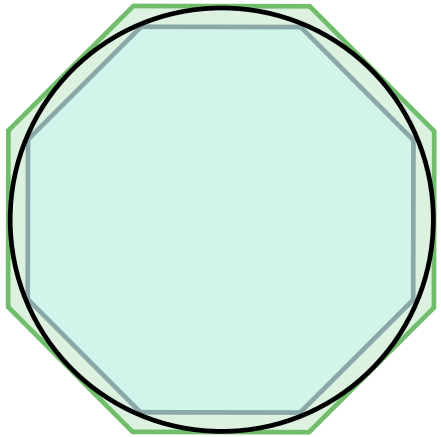
Cf. http://www.cs.utep.edu/interval-comp/, click on *Early papers by Others*.

Archimedes and an enclosure of π









Historical remarks

Childhood until the seventies.

Popularization in the 1980, German school (U. Kulisch).

IEEE-754 standard for floating-point arithmetic in 1985:
directed roundings are standardized and available (?).

Since the nineties: interval **algorithms**.

IEEE-1788 standard for interval arithmetic in 2014?
I hope so...

Agenda

Definitions: operations

$\mathbf{x} \diamond \mathbf{y} = \mathbf{Hull}\{x \diamond y \ : \ x \in \mathbf{x}, y \in \mathbf{y}\}$

Arithmetic and algebraic operations: use the monotonicity

$$
\begin{array}{lcl}
[\underline{x}, \overline{x}] + [\underline{y}, \overline{y}] & = & [\underline{x} + \underline{y}, \overline{x} + \overline{y}] \\
[\underline{x}, \overline{x}] - [\underline{y}, \overline{y}] & = & [\underline{x} - \overline{y}, \overline{x} - \underline{y}] \\
[\underline{x}, \overline{x}] \times [\underline{y}, \overline{y}] & = & [\min(\underline{x} \times \underline{y}, \underline{x} \times \overline{y}, \overline{x} \times \underline{y}, \overline{x} \times \overline{y}), \max(\text{ibid.})] \\
[\underline{x}, \overline{x}]^2 & = & [\min(\underline{x}^2, \overline{x}^2), \max(\underline{x}^2, \overline{x}^2)] \ \text{ if } 0 \notin [\underline{x}, \overline{x}] \\
 & & [0, \max(\underline{x}^2, \overline{x}^2)] \ \text{ otherwise}
\end{array}
$$

Interval arithmetic: implementation using floating-point arithmetic

Implementation using floating-point arithmetic:
use directed rounding modes (cf. IEEE 754 standard)

$$\sqrt{[2,3]} = [\nabla\sqrt{2}, \triangle\sqrt{3}]$$

Advantage: every result is guaranteed, in the sense that the exact, unknown result, belongs to the computed interval result.

Operations

Algebraic properties: associativity, commutativity hold, some are lost:

- subtraction is not the inverse of addition, in particular $\mathbf{x} - \mathbf{x} \neq [0]$
- division is not the inverse of multiplication
- squaring is tighter than multiplication by oneself
- multiplication is only sub-distributive wrt addition
- with floating-point implementation, operations are not associative either

Definitions: comparisons

How to compare two intervals?

how to compare $[-1, 2]$ and $[0, 3]$? or $[-1, 2]$ and $[0, 1]$?

Several approaches:

- use explicit names: CertainlyLess, PossiblyLess
- use trivalued logic (MPFI): $\mathbf{a} < \mathbf{b}$ returns
  - -1 if every element of $\mathbf{a}$ is $<$ than every element of $\mathbf{b}$,
  - $+1$ if every element of $\mathbf{a}$ is $>$ than every element of $\mathbf{b}$,
  - 0 if $\mathbf{a}$ and $\mathbf{b}$ overlap.
- use many more relation names, cf. IEEE 1788.

IEEE-1788 standard: comparison relations

- **7 relations:** equal ($=$), subset ($\subset$), less than or equal to ($\leq$), precedes or touches ($\preceq$), interior to, less than ($<$), precedes ($\prec$).

IEEE-1788 standard: comparison relations

- **7 relations:** equal ($=$), subset ($\subset$), less than or equal to ($\leq$), precedes or touches ($\preceq$), interior to, less than ($<$), precedes ($\prec$).
- **Interval overlapping relations:** before, meets, overlaps, starts, containedBy, finishes, equal, finishedBy, contains, startedBy, overlappedBy, metBy, after.

  Again, relations defined by conditions on the bounds.

Agenda

Definitions: function extension

Definition:

an interval extension **f** of a function f satisfies

$$\forall \mathbf{x},\ f(\mathbf{x}) \subset \mathbf{f}(\mathbf{x}),\ \text{and}\ \forall x,\ f(\{x\}) = \mathbf{f}(\{x\}).$$

Elementary functions: again, use the monotony.

$$\begin{array}{lcl} \exp \mathbf{x} & = & [\exp \underline{x}, \exp \overline{x}] \\ \log \mathbf{x} & = & [\log \underline{x}, \log \overline{x}] \ \text{if}\ \underline{x} \geq 0, [-\infty, \log \overline{x}] \ \text{if}\ \overline{x} > 0 \\ \sin[\pi/6, 2\pi/3] & = & [1/2, 1] \\ \ldots & & \end{array}$$

Definitions: function extension

Example: $f(x) = x^2 - x + 1$ with $x \in [-2, 1]$.
$[-2,1]^2 - [-2,1] + 1 = [0,4] + [-1,2] + 1 = [0,7]$.
Since $x^2 - x + 1 = x(x-1) + 1$, we get $[-2,1] \cdot ([-2,1] - 1) + 1 = [-2,1] \cdot [-3,0] + 1 = [-3,6] + 1 = [-2,7]$.
Since $x^2 - x + 1 = (x - 1/2)^2 + 3/4$, we get $([-2,1] - 1/2)^2 + 3/4 = [-5/2, 1/2]^2 + 3/4 = [0, 25/4] + 3/4 = [3/4, 7] = f([-2,1])$.

Problem with this definition: infinitely many interval extensions, syntactic use (instead of semantic).

How to choose the best extension? How to choose a good one?

Definitions: function extension

Mean value theorem of order 1 (Taylor expansion of order 1):

$\forall x, \forall y, \exists \xi_{x,y} \in (x,y) \ : \ f(y) = f(x) + (y-x) \cdot f'(\xi_{x,y})$

Interval interpretation:

$\forall y \in \mathbf{x}, \forall \tilde{x} \in \mathbf{x}, \ f(y) \in f(\tilde{x}) + (y - \tilde{x}) \cdot \mathbf{f'}(\mathbf{x})$

$\Rightarrow f(\mathbf{x}) \subset f(\tilde{x}) + (\mathbf{x} - \tilde{x}) \cdot \mathbf{f'}(\mathbf{x})$

Mean value theorem of order 2 (Taylor expansion of order 2):

$\forall x, \forall y, \exists \xi_{x,y} \in (x,y) : f(y) = f(x) + (y-x) \cdot f'(x) + \frac{(y-x)^2}{2} \cdot f''(\xi_{x,y})$

Interval interpretation:

$\forall y \in \mathbf{x}, \forall \tilde{x} \in \mathbf{x}, \ f(y) \in f(\tilde{x}) + (y - \tilde{x}) \cdot f'(\tilde{x}) + \frac{(y-\tilde{x})^2}{2} \cdot \mathbf{f''}(\mathbf{x})$

$\Rightarrow f(\mathbf{x}) \subset f(\tilde{x}) + (\mathbf{x} - \tilde{x}) \cdot f'(\tilde{x}) + \frac{(\mathbf{x}-\tilde{x})^2}{2} \cdot \mathbf{f''}(\mathbf{x}).$

Definitions: function extension

No need to go further:

- it is difficult to compute (automatically) the derivatives of higher order,
  especially for multivariate functions;
- there is no (theoretical) gain in quality.

Theorem:

- for the natural extension $\mathbf{f}$ of f, it holds
  $d(f(\mathbf{x}), \mathbf{f}(\mathbf{x})) \leq \mathcal{O}(w(\mathbf{x}))$
- for the first order Taylor extension $\mathbf{f}_{\mathbf{T}_1}$ of f, it holds
  $d(f(\mathbf{x}), \mathbf{f}_{\mathbf{T}_1}(\mathbf{x})) \leq \mathcal{O}(w(\mathbf{x})^2)$
- getting an order higher than 3 is impossible without the squaring operation, is difficult even with it. . .

Agenda

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).

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) $\mathbf{I}$ such that $\forall x \in [-1,1],\ f(x) \in p(x) + \mathbf{I}$.

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.

Agenda

Definitions: intervals, vectors, matrices

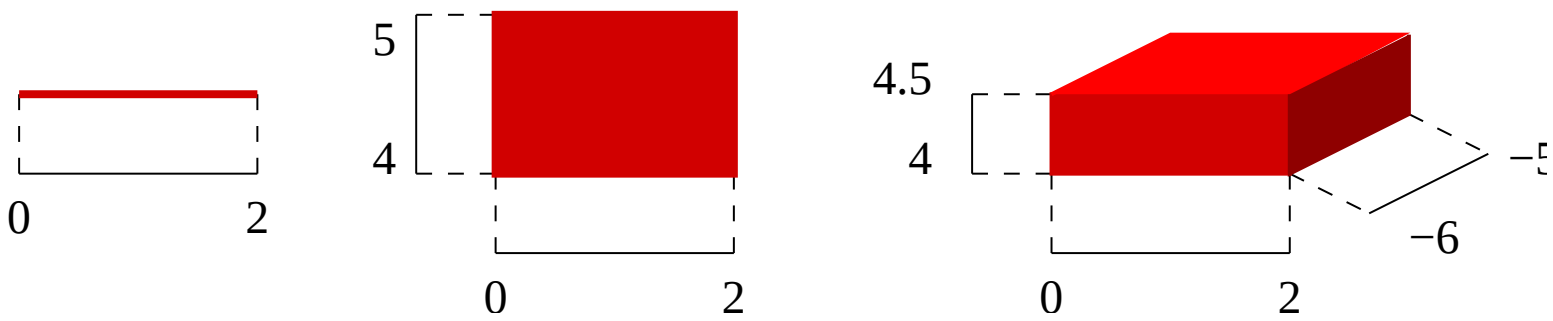
Objects:

- intervals of real numbers = closed connected sets of $\mathbb{R}$
  - interval for π: $[3.14159, 3.14160]$
  - data d measured with an absolute error less than $\pm\varepsilon$: $[d-\varepsilon, d+\varepsilon]$
- interval vector: components = intervals; also called *box*

$[0\ ;\ 2]$

$\begin{pmatrix}[0\ ;\ 2]\\ [4\ ;\ 5]\end{pmatrix}$

$\begin{pmatrix}[0;2]\\ [4\ ;\ 4.5]\\ [-6\ ;\ -5]\end{pmatrix}$



- interval matrix: components = intervals.

Agenda

Cons: overestimation (1/2)

The result encloses the true result, but it is too large:
overestimation phenomenon.
Two main sources: variable dependency and wrapping effect.

(Loss of) Variable dependency:

$$\mathbf{x} - \mathbf{x} = \{x - y \ : \ x \in \mathbf{x}, y \in \mathbf{x}\} \neq \{x - x \ : \ x \in \mathbf{x}\} = \{0\}.$$

Cons: overestimation (2/2)

Wrapping effect

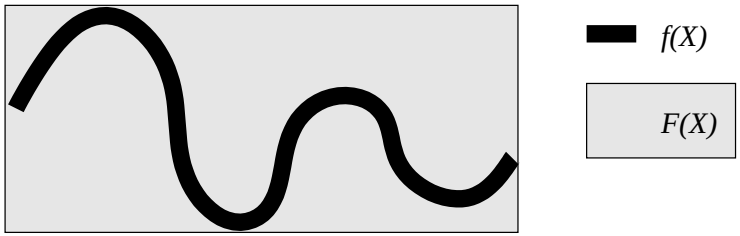
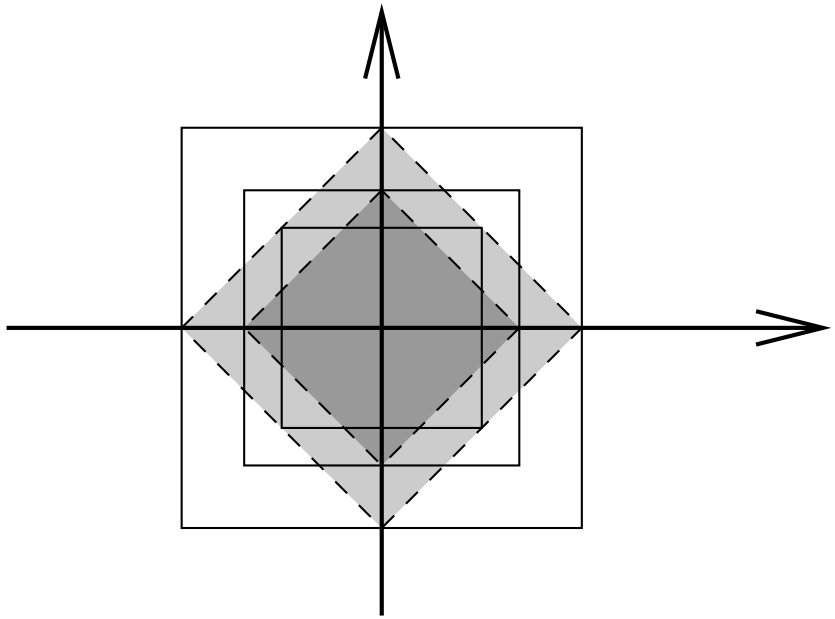


image of $f(\mathbf{x})$
with $f : \mathbb{R}^2 \to \mathbb{R}^2$



2 successive rotations of $\pi/4$
of the little central square

Cons: complexity and efficiency

Complexity: most problems are NP-hard **(Gaganov, Rohn, Kreinovich...)**

- evaluate a function on a box... even up to ε
- solve a linear system... even up to $1/4n^4$
- determine if the solution of a linear system is bounded

Efficiency

Implementation using floating-point arithmetic:

use directed roundings, towards $\pm\infty$.

Overhead in execution time:

in theory, at most 4, or 8, cf.

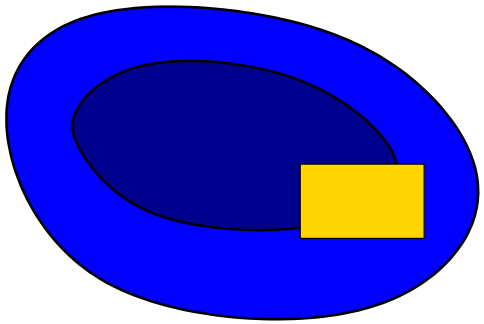
$$[\underline{x}, \overline{x}] \times [\underline{y}, \overline{y}] = [\begin{array}{l} \min(\mathrm{RD}(\underline{x} \times \underline{y}), \mathrm{RD}(\underline{x} \times \overline{y}), \mathrm{RD}(\overline{x} \times \underline{y}), \mathrm{RD}(\overline{x} \times \overline{y})), \\ \max(\mathrm{RU}(\underline{x} \times \underline{y}), \mathrm{RU}(\underline{x} \times \overline{y}), \mathrm{RU}(\overline{x} \times \underline{y}), \mathrm{RU}(\overline{x} \times \overline{y})) \end{array}$$

in practice, around 20: changing the rounding modes implies flushing the pipelines (on most architectures and implementations).

Agenda

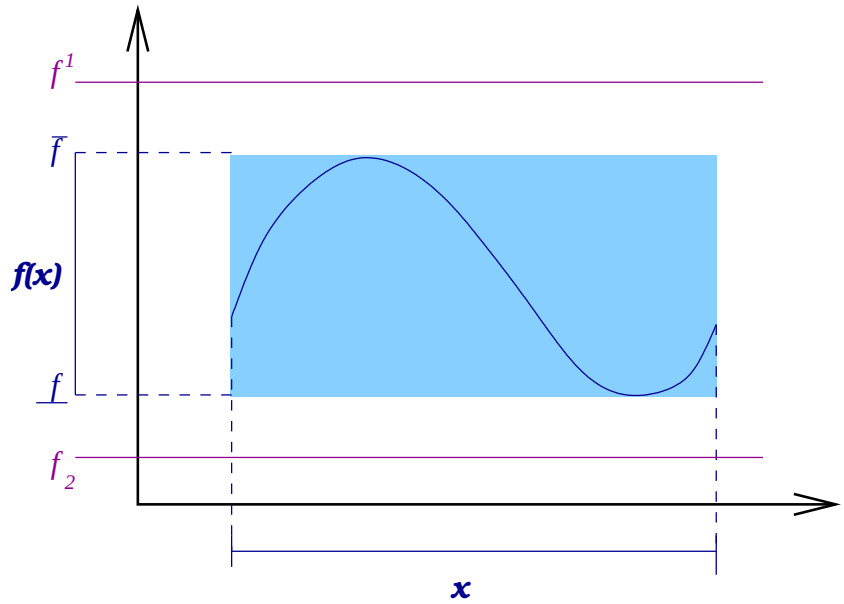
Pros: set computing

Behaviour safe? controllable? dangerous?



always controllable.

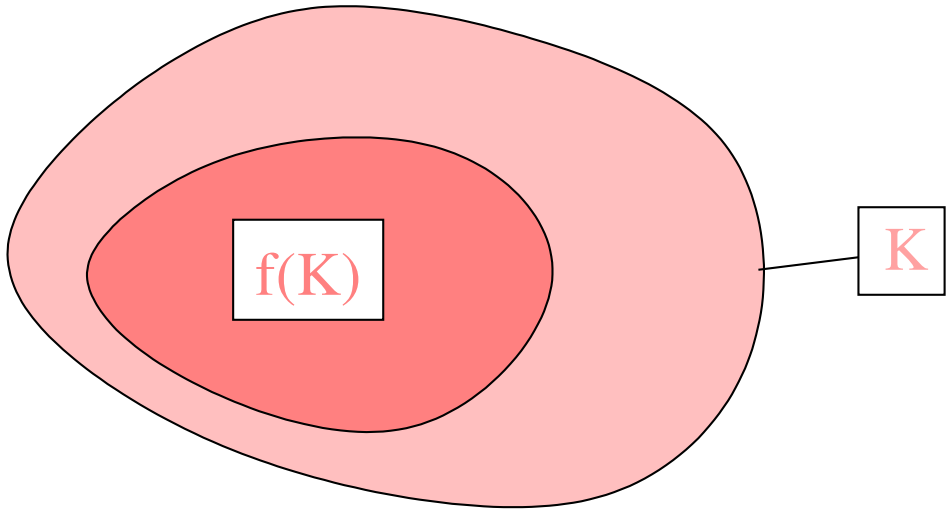
On **x**, are the extrema of the function f $> f^1$, $< f_2$?



No if $f(\mathbf{x}) = [\underline{f}, \overline{f}] \subset [f_2, f^1]$.

Pros: Brouwer-Schauder theorem

A function f which is continuous on the unit ball B and which satisfies $f(B) \subset B$ has a fixed point on B.
Furthermore, if $f(B) \subset \text{int}B$ (and some other conditions) then f has a unique fixed point on B.



The theorem remains valid if B is replaced by a compact K and in particular an interval.

Agenda

Algorithm: solving a nonlinear system: Newton

Why a specific iteration for interval computations?

Usual formula:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

Direct interval transposition:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \frac{f(\mathbf{x}_k)}{f'(\mathbf{x}_k)}$$

Algorithm: solving a nonlinear system: Newton

Why a specific iteration for interval computations?

Usual formula:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

Direct interval transposition:

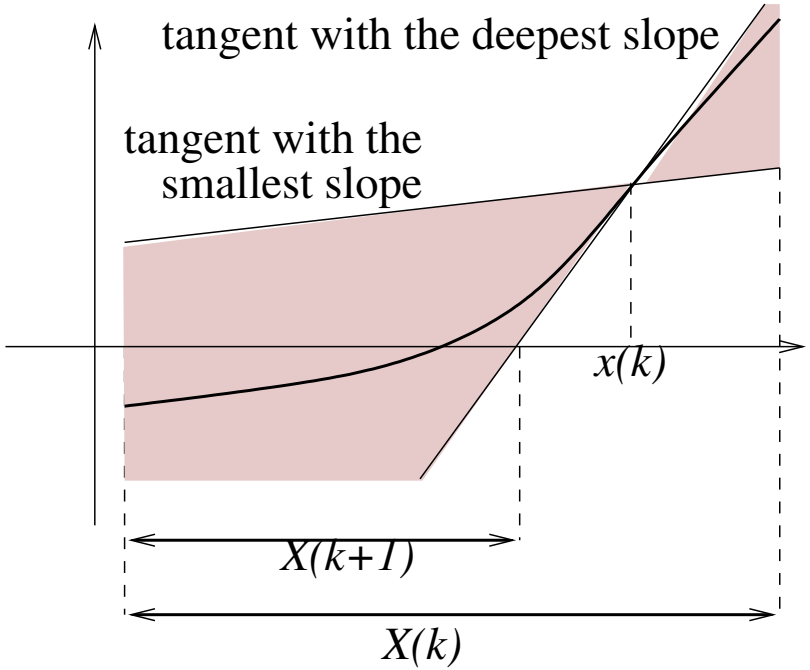
$$\mathbf{x}_{k+1} = \mathbf{x}_k - \frac{f(\mathbf{x}_k)}{f'(\mathbf{x}_k)}$$

Width of the resulting interval:

$$w(\mathbf{x}_{k+1}) = w(\mathbf{x}_k) + w\left(\frac{f(\mathbf{x}_k)}{f'(\mathbf{x}_k)}\right) > w(\mathbf{x}_k)$$

divergence!

Algorithm: interval Newton (Hansen-Greenberg 83, Baker Kearfott 95-97, Mayer 95, van Hentenryck et al. 97)

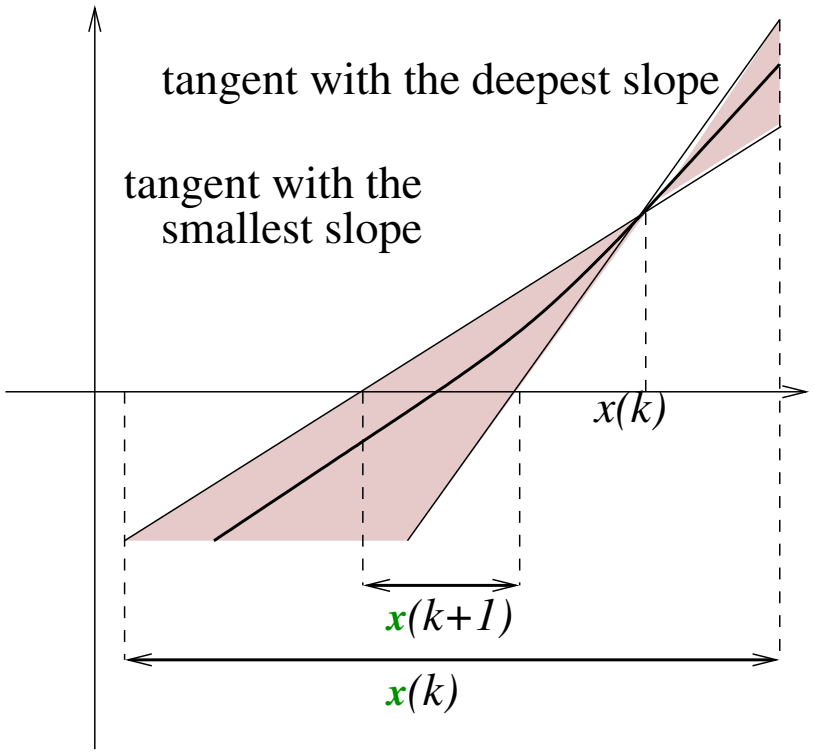


$$\mathbf{x}_{k+1} := \left(x_k - \frac{\mathbf{f}(\{x_k\})}{\mathbf{f}'(\mathbf{x}_k)} \right) \bigcap \mathbf{x}_k$$

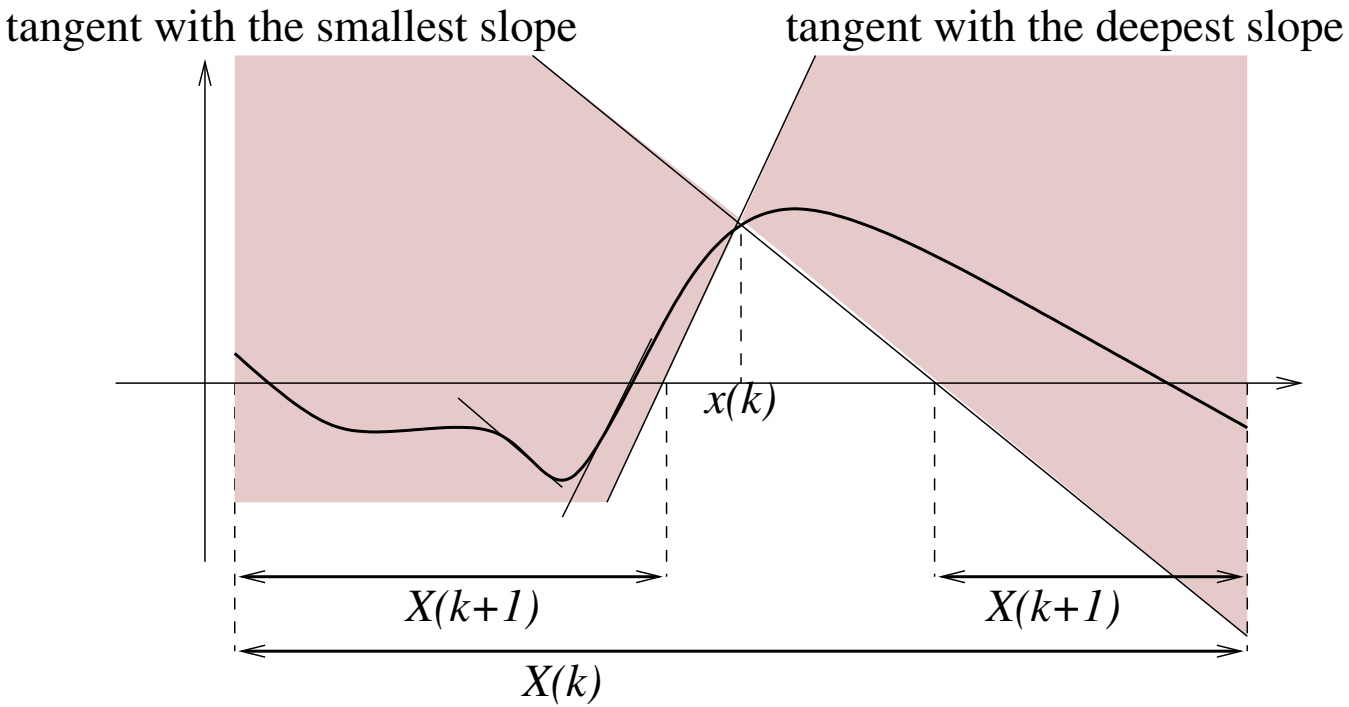
Interval Newton: Brouwer theorem

If the new iterate (before intersection) is a subset of the previous iterate, then f has a zero on it.

Furthermore, if it is included in its interior, then this zero is unique.



Algorithm: interval Newton



$$(\mathbf{x}_{k+1,1}, \mathbf{x}_{k+1,2}) := \left(x_k - \frac{\mathbf{f}(\{x_k\})}{\mathbf{f}'(\mathbf{x}_k)} \right) \bigcap \mathbf{x}_k$$

Algorithm: interval Newton

Input: $\mathbf{f}$, $\mathbf{f}'$, $\mathbf{x}_0$ // $\mathbf{x}_0$ initial search interval

Initialization: $\mathcal{L} = \{\mathbf{x}_0\}$, $\alpha = 0.75$ //any value in $]0.5, 1[$ is suitable

Loop: while $\mathcal{L} \neq \emptyset$

Suppress $(\mathbf{x}, \mathcal{L})$

$x := \text{mid}(\mathbf{x})$

$(\mathbf{x}_1, \mathbf{x}_2) := \left(x - \frac{\mathbf{f}(\{x\})}{\mathbf{f}'(\mathbf{x})}\right) \bigcap \mathbf{x}$ // $\mathbf{x}_1$ and $\mathbf{x}_2$ can be empty

if $w(\mathbf{x}_1) > \alpha w(\mathbf{x})$ or $w(\mathbf{x}_2) > \alpha w(\mathbf{x})$ then $(\mathbf{x}_1, \mathbf{x}_2) := \text{bisect}(\mathbf{x})$

if $\mathbf{x}_1 \neq \emptyset$ and $\mathbf{f}(\mathbf{x}_1) \ni 0$ then

if $w(\mathbf{x}_1)/|\text{mid}(\mathbf{x}_1)| \leq \varepsilon_{\mathbf{x}}$ or $w(\mathbf{f}(\mathbf{x}_1)) \leq \varepsilon_Y$ then Insert $\mathbf{x}_1$ in Res

else Insert $\mathbf{x}_1$ in $\mathcal{L}$

same handling of $\mathbf{x}_2$

Output: *Res*, a list of intervals that may contain the roots.

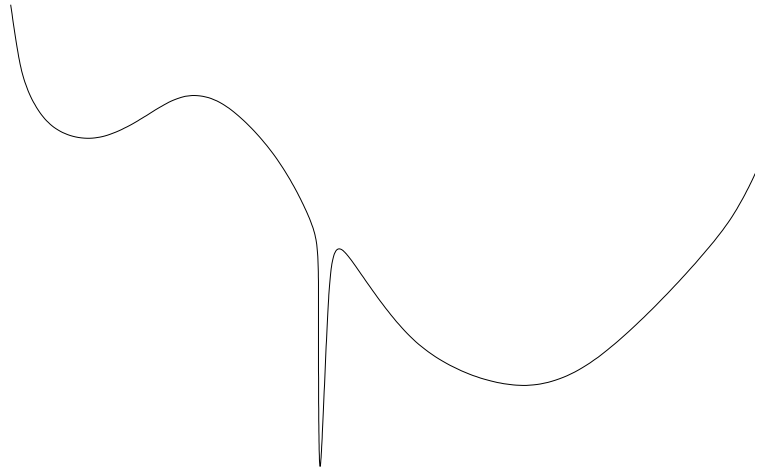
Algorithm: optimize a continuous function

Problem: $f : \mathbf{R}^n \to \mathbf{R}$, determine x^* and f^* that verify

$$f^* = f(x^*) = \min_x f(x)$$

Assumptions:

- search within a box $\mathbf{x}_0$
- $x^* \in$ in the interior of $(\mathbf{x}_0)$, not at the boundary
- f continuous enough: $\mathcal{C}^2$



Algorithm: optimize a continuous function

(Ratschek and Rokne 1988, Hansen 1992, Kearfott 1996...)

Goal: find the minimum of f, continuous function on a box $\mathbf{x}_0$.

$\mathbf{x}_0$ current box

$\bar{f}$ current upper bound of f^*

while there is a box in the waiting list

if $f(\mathbf{x}) > \bar{f}$ then

reject $\mathbf{x}$

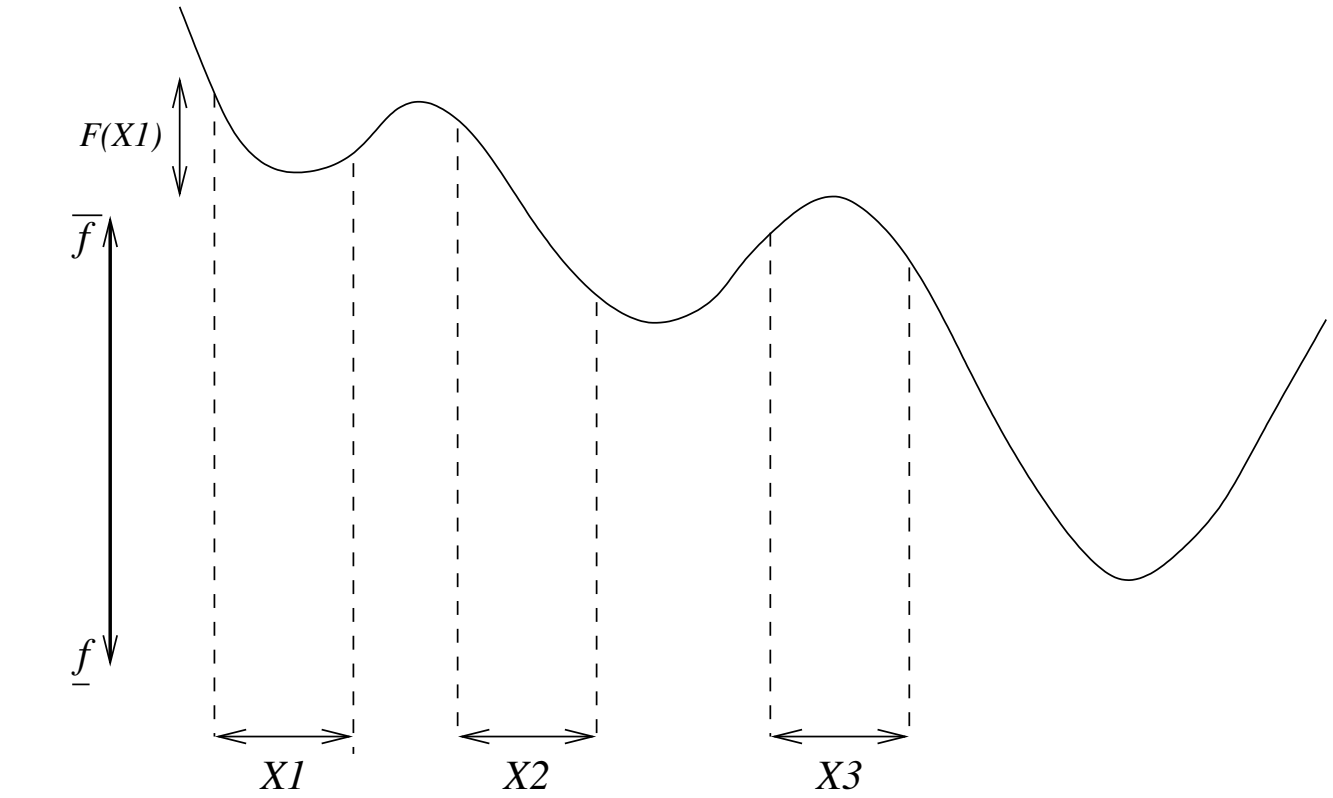
otherwise

update $\bar{f}$: if $f(mid(\mathbf{x})) < \bar{f}$ then $\bar{f} = f(mid(\mathbf{x}))$

bisect $\mathbf{x}$ into $\mathbf{x}_1$ and $\mathbf{x}_2$

examine $\mathbf{x}_1$ and $\mathbf{x}_2$

Algorithm: optimize a continuous function
the rejection procedure

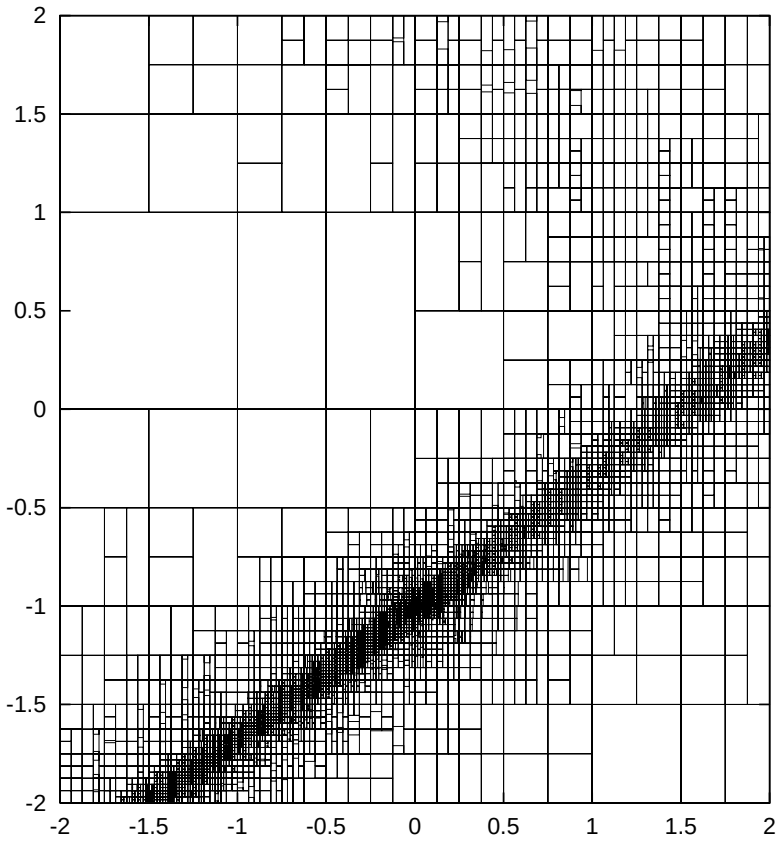


Algorithm: optimize a continuous function

Hansen algorithm Hansen 1992

$\mathcal{L}$ = list of not yet examined boxes := $\{\mathbf{x}_0\}$
while $\mathcal{L} \neq \emptyset$ **loop**
    remove $\mathbf{x}$ from $\mathcal{L}$
    reject $\mathbf{x}$**?**
        yes if $f(\mathbf{x}) > \overline{f}$
        yes if $\mathrm{Grad} f(\mathbf{x}) \not\ni 0$
        yes if $Hf(\mathbf{x})$ has its diagonal non > 0
    reduce $\mathbf{x}$
        Newton applied to the gradient
        solve $\mathbf{y} \subset \mathbf{x}$ such that $f(\mathbf{y}) \leq \overline{f}$
    bisect $\mathbf{y}$: insert the resulting $\mathbf{y}_1$ and $\mathbf{y}_2$ in $\mathcal{L}$

Example of the splitting of the box $[-2,2]^2$



Agenda

Precious features

- **Fundamental theorem of interval arithmetic ("Thou shalt not lie"):** the returned result contains the sought result;
- **Brouwer theorem:** proof of existence (and uniqueness) of a solution;
- **ad hoc division:** gap between two semi-infinite intervals is preserved.

Precious features

- **Fundamental theorem of interval arithmetic ("Thou shalt not lie"):** the returned result contains the sought result;
- **Brouwer theorem:** proof of existence (and uniqueness) of a solution;
- **ad hoc division:** gap between two semi-infinite intervals is preserved.

Goal of a standardization: keep the nice properties, have common definitions.

IEEE P1788 working group

Creation of the IEEE P1788 project: Initiated by 15 attenders at Dagstuhl, Jan 2008. Project authorised as IEEE-WG-P1788, Jun 2008, until December 2014.

How P1788's work is done

- The bulk of work is carried out by email - electronic voting.
- Motions are proposed, seconded; three weeks discussion period; three weeks voting period.
- IEEE has given us a four year deadline + two more: December 2014.
- One "in person" meeting per year is planned — next one during IFSA/NAFIPS 2013, June, Edmonton, Canada.
- IEEE auspices: 1 report + 1 teleconference quarterly

IEEE P1788 working group

Creation of the IEEE P1788 project: Initiated by 15 attenders at Dagstuhl, Jan 2008. Project authorised as IEEE-WG-P1788, Jun 2008, until December 2014.

How P1788's work is done

- The bulk of work is carried out by email - electronic voting.
- Motions are proposed, seconded; three weeks discussion period; three weeks voting period.
- IEEE has given us a four year deadline + two more: December 2014.
- One "in person" meeting per year is planned — next one during IFSA/NAFIPS 2013, June, Edmonton, Canada.
- IEEE auspices: 1 report + 1 teleconference quarterly

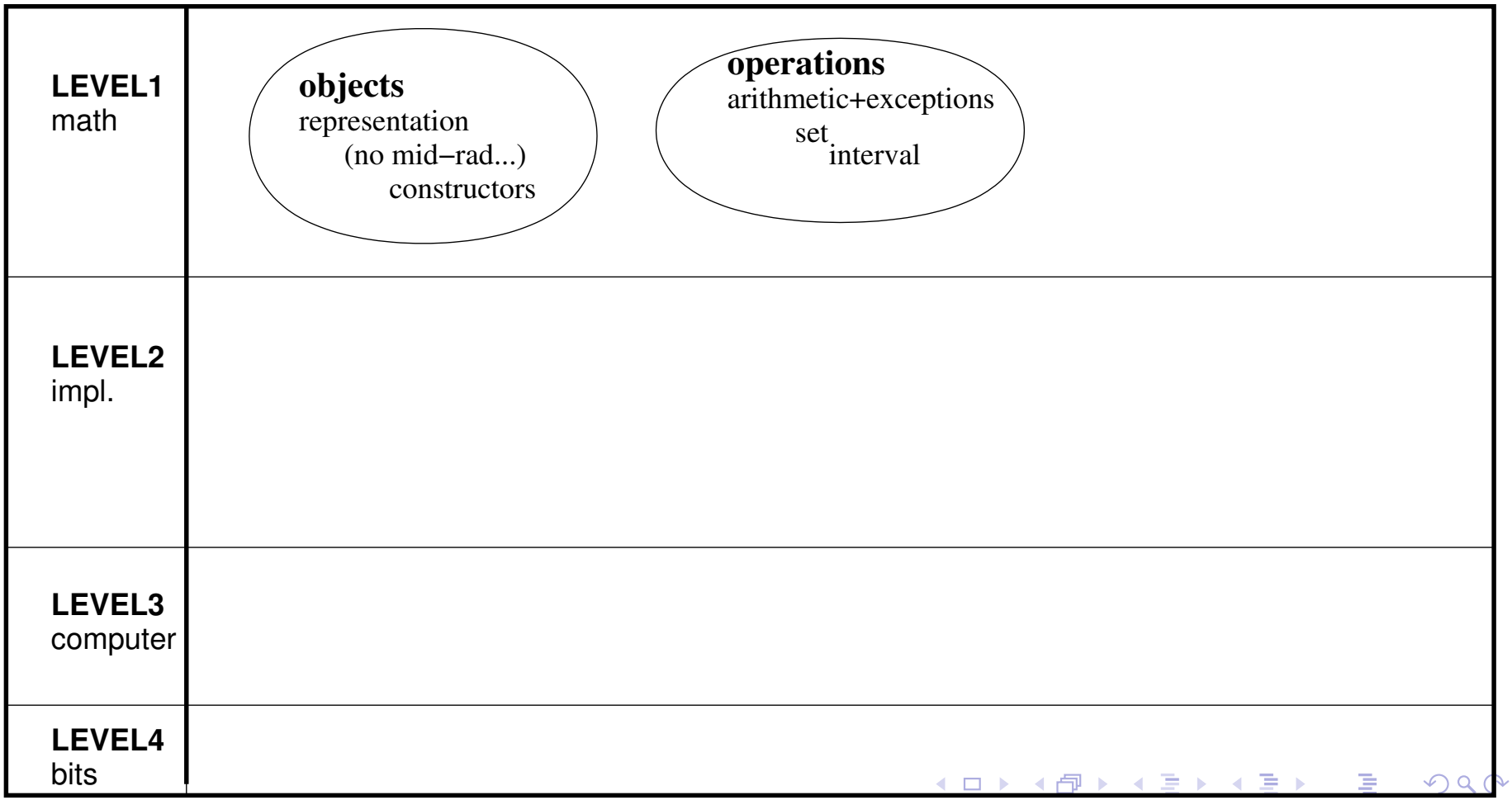
IEEE-1788 standard: the big picture

LEVEL1 math	
LEVEL2 impl.	
LEVEL3 computer	
LEVEL4 bits	

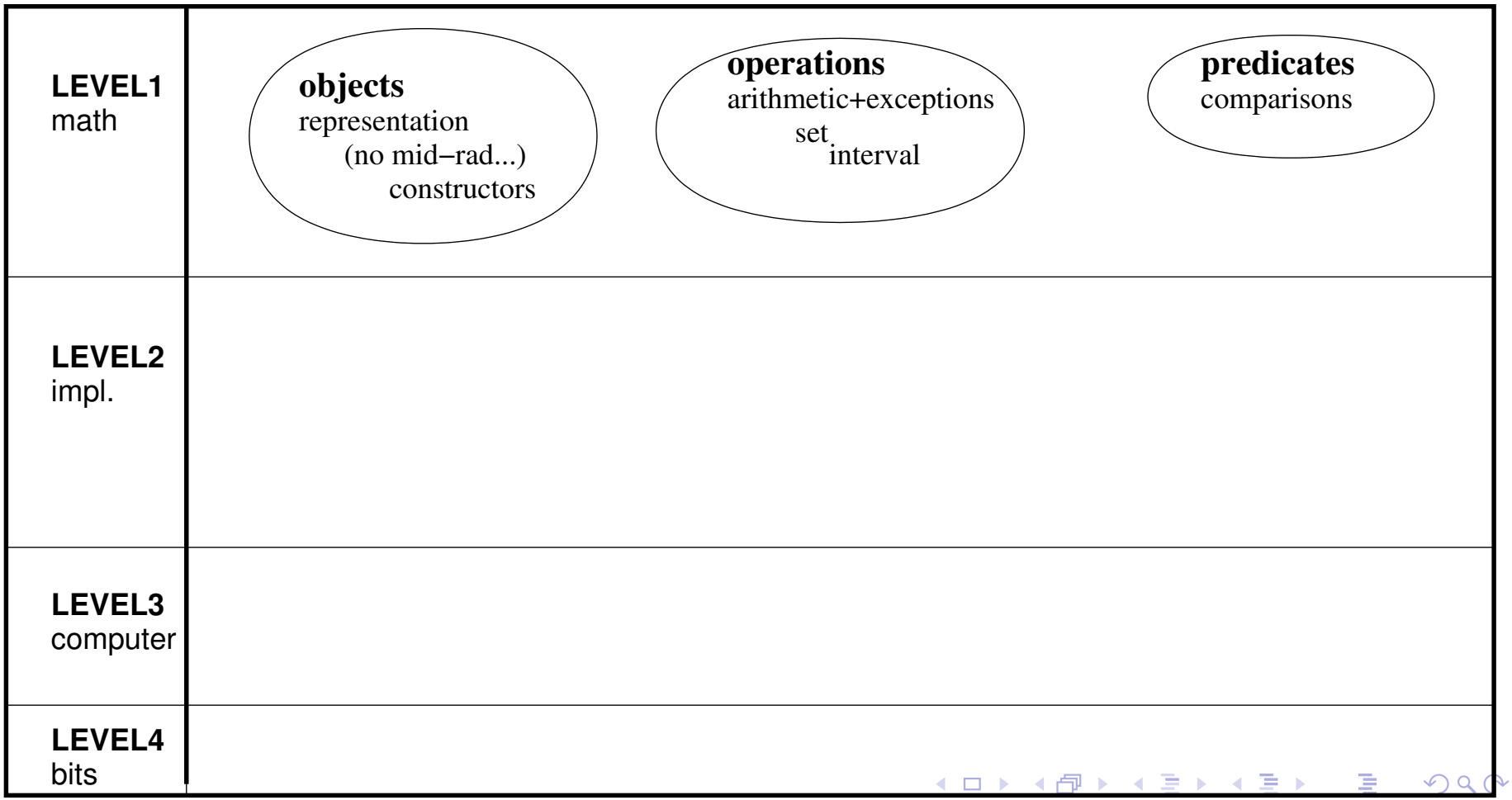
IEEE-1788 standard: the big picture

LEVEL1 math	**objects** representation (no mid–rad...) constructors
LEVEL2 impl.	
LEVEL3 computer	
LEVEL4 bits	

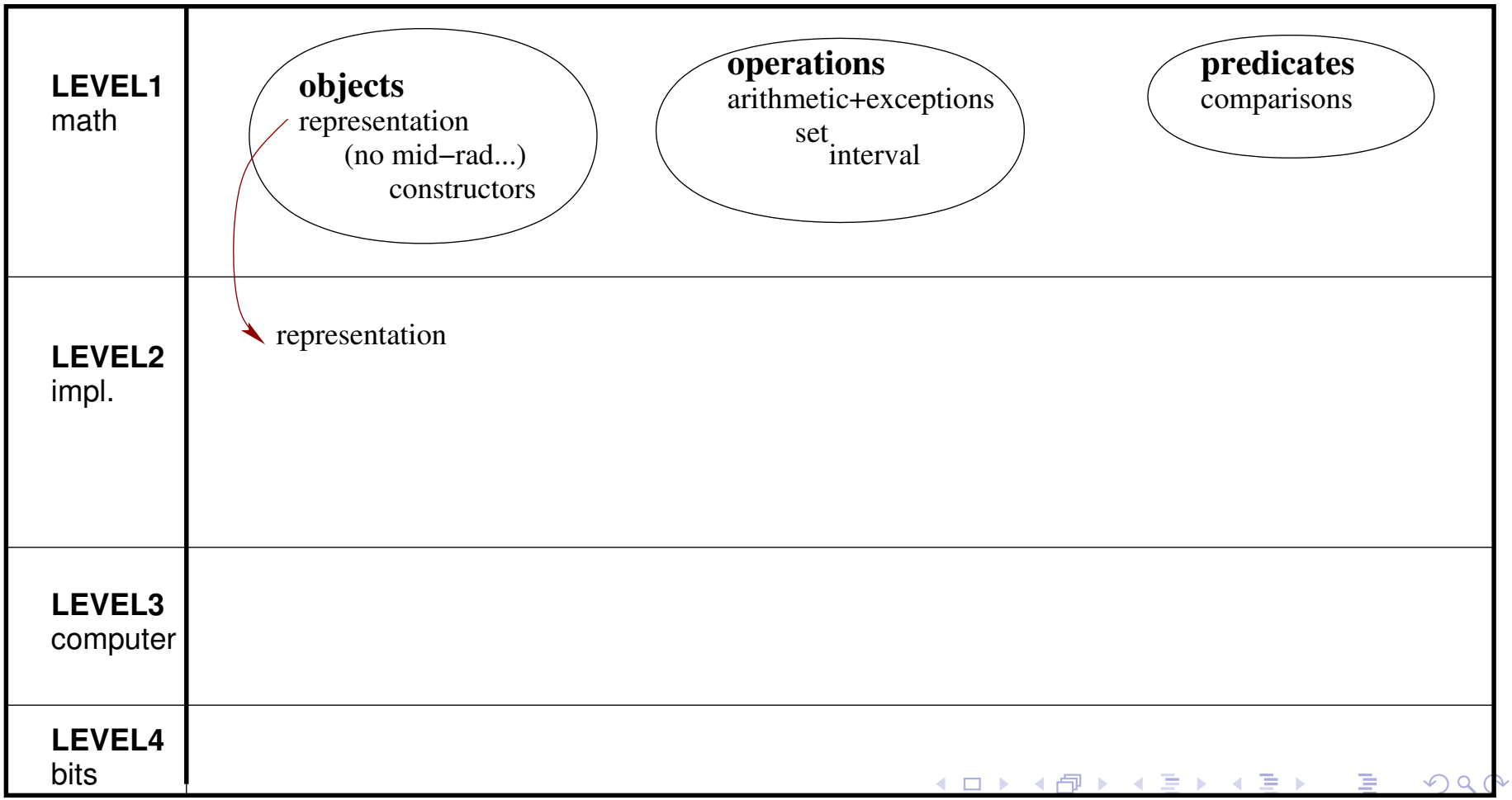
IEEE-1788 standard: the big picture



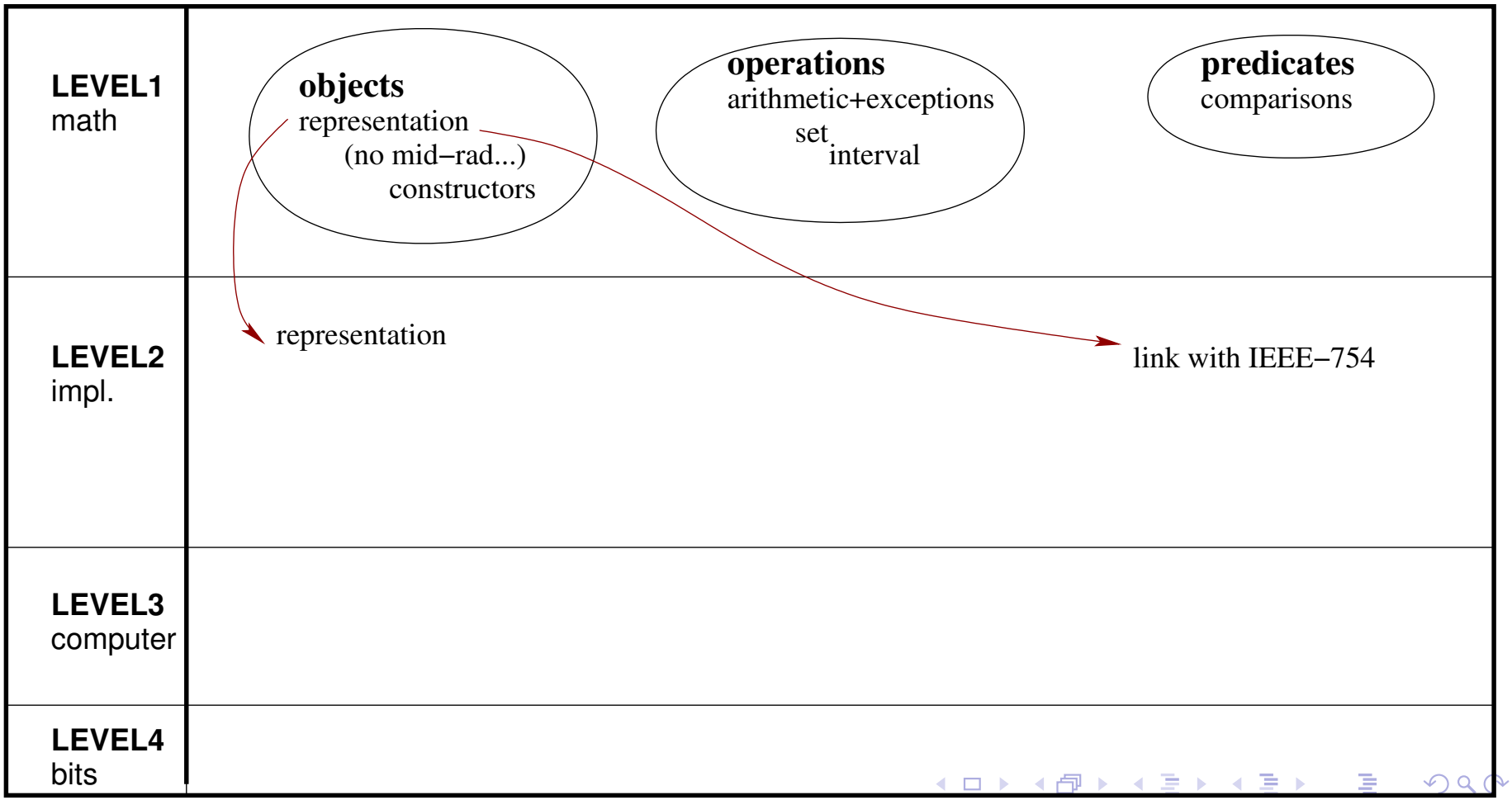
IEEE-1788 standard: the big picture



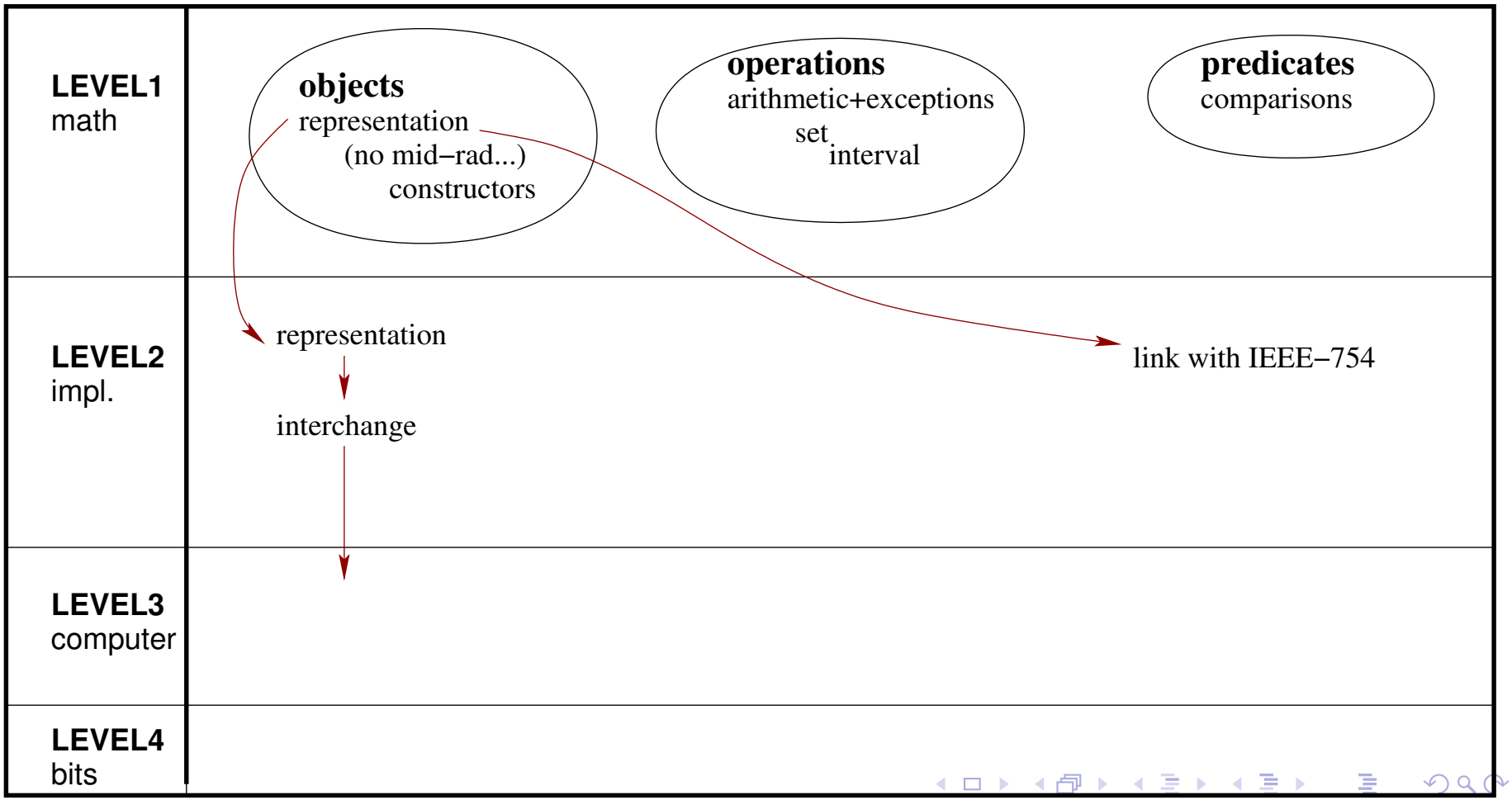
IEEE-1788 standard: the big picture



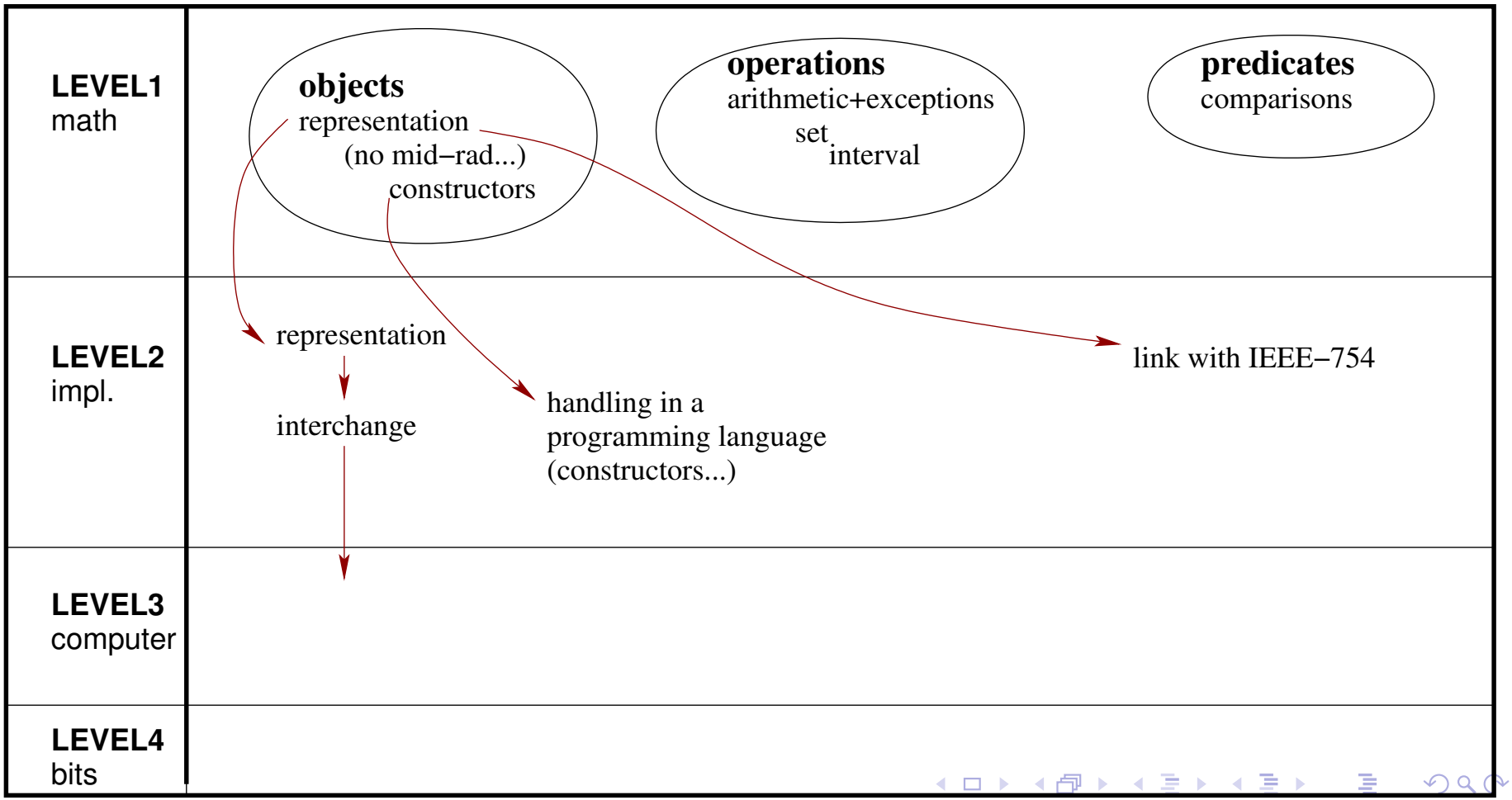
IEEE-1788 standard: the big picture



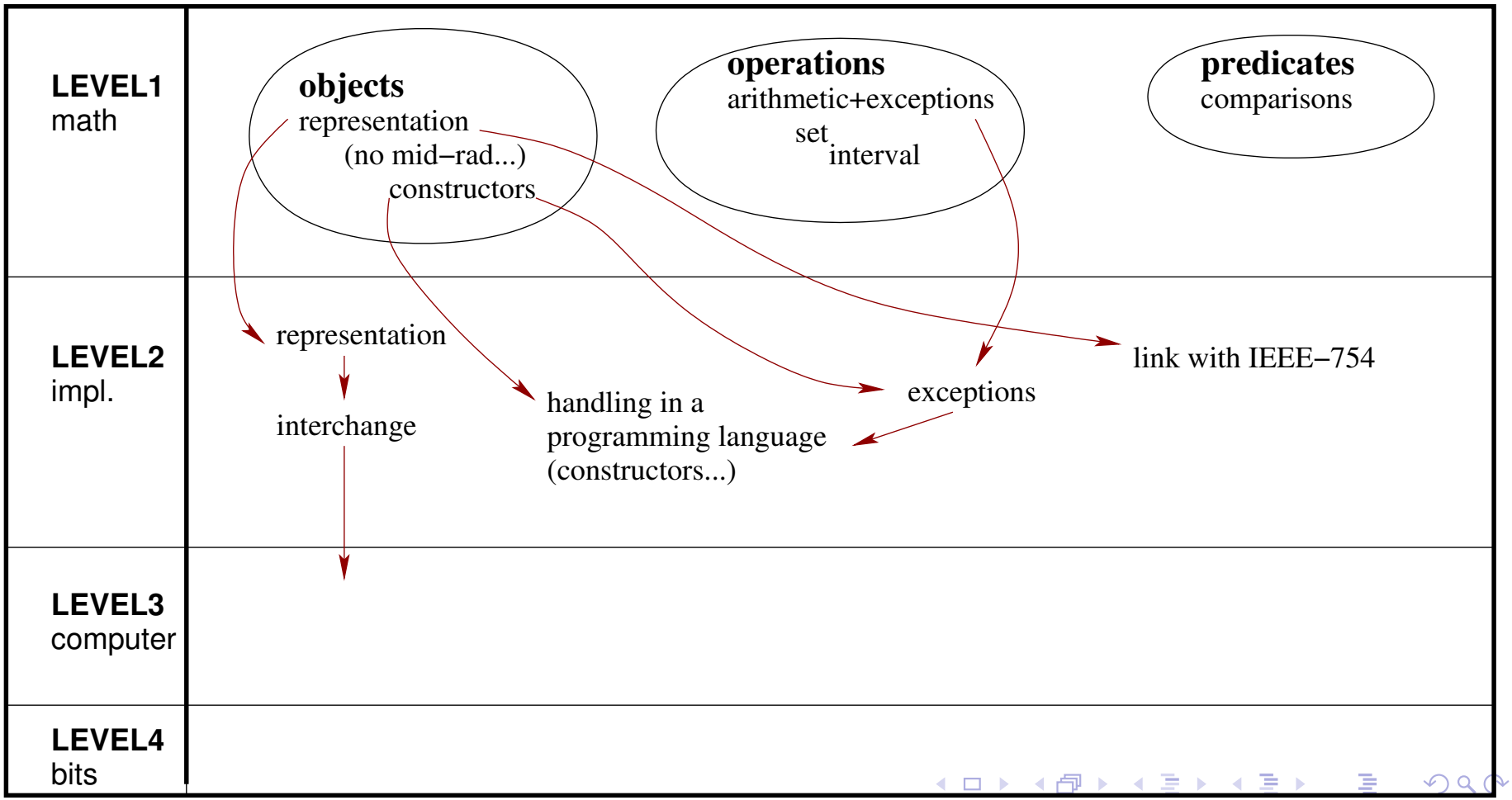
IEEE-1788 standard: the big picture



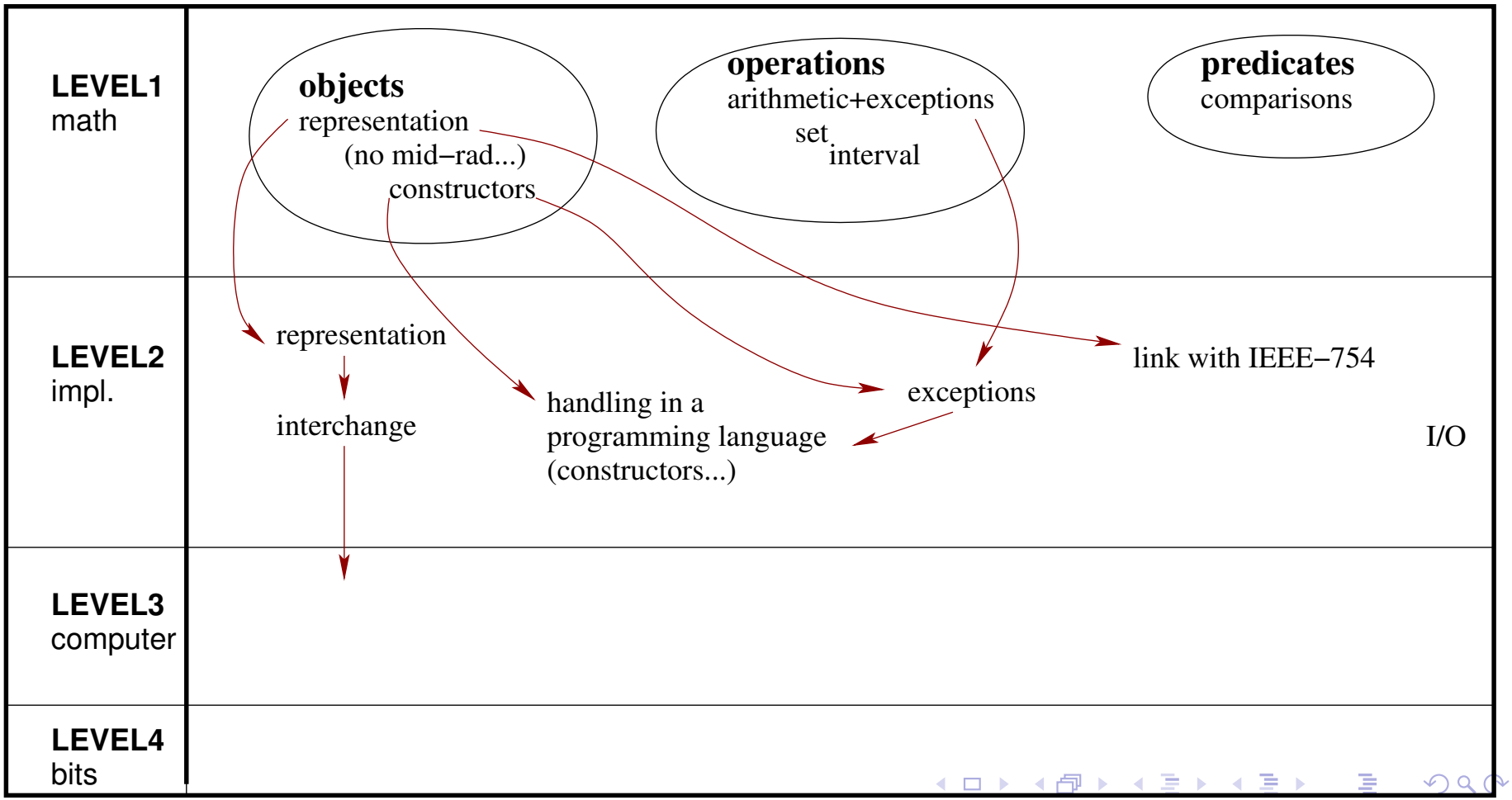
IEEE-1788 standard: the big picture



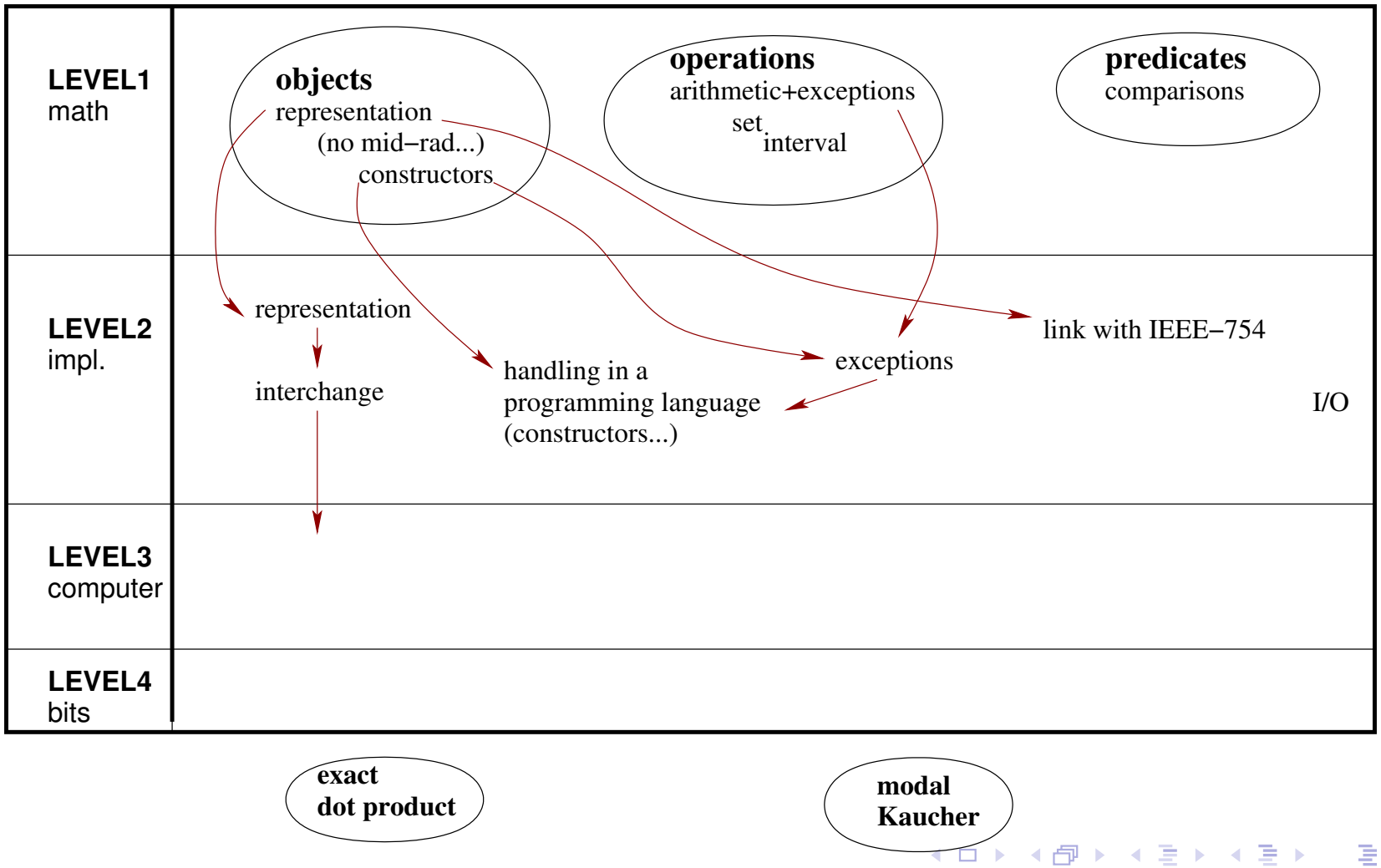
IEEE-1788 standard: the big picture



IEEE-1788 standard: the big picture



IEEE-1788 standard: the big picture



Agenda

Repeatability and reproducibility

Repeatability:
getting the same result (the same bits) from run to run, on the same machine.

Reproducibility:
getting the same result (the same bits) from run to run, whatever the machine.

Numerical reproducibility?

Definition?

- Numerical reproducibility = best possible result = correct rounding of the exact result?
- Numerical reproducibility = getting the same string of bits whatever the run?

Cf. Nguyen and Demmel (Arith 2013): implementations of the second choice for the summation, with a tradeoff between the accuracy of the result and the execution time.

New light on numerical reproducibility:

- reproducibility and correct rounding are separate notions
- a hierarchy of reproducibility levels exists: accuracy vs execution time.

Reproducibility: influence of the computing precision (1/2)

The computing precision may depend on

- the processor (Binary32, Binary64, 80bits-registers),
- the language (Fortran vs C, cf. Florent's talk).

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

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

Examples: linear systems solving, Newton iteration.

Influence of the expression

Using floating-point arithmetic: the problem comes from the non-associativity of the operations

$$(a_1 + a_2) + (a_3 + a_4) \neq ((a_1 + a_2) + a_3) + a_4.$$

Using interval arithmetic: the expression influences the result because operations are neither distributive nor reciprocal ($+$ of $-$, $\times$ of $/$).

Using interval arithmetic implemented with floating-point arithmetic: because operations are neither distributive nor reciprocal ($+$ of $-$, $\times$ of $/$) nor associative: problems cumulate.

Influence of the expression: example

$$[2^{100}, 2^{100}] + [1, 1] - [2^{100}, 2^{100}]?$$

With these parentheses:

$$([2^{100}, 2^{100}]+[1, 1])-[2^{100}, 2^{100}] = [2^{100}, \mathrm{succ}(2^{100})]-[2^{100}, 2^{100}] = [0, \mathrm{ulp}(2^{100})].$$

With those parentheses:

$$([2^{100}, 2^{100}] - [2^{100}, 2^{100}]) + [1, 1] = [0, 0] + [1, 1] = [1, 1].$$

Both include the results, one is more accurate than the other...

Moral lesson: interval results are always guaranteed to include the exact result, whatever the chosen expression. However their accuracy strongly depends on the chosen expression, on the order of operations.

More on the influence of the order of the operations

Beware "hidden" assumptions on the order of the operations.

Example: interval matrix product.
In order to save 1 or 2 calls to `gemm` (BLAS matrix product), Rump's algorithm (2012) assumes that $\mathbf{A}_m \cdot \mathbf{B}_m$ and $|\mathbf{A}_m| \cdot |\mathbf{B}_m|$ are computed in the same order.

BLAS do not guarantee anything on the order of operations nor on the reproducibility of this order from one product to the next.

Moral lesson: interval results could depend on the order of operations,
interval results could be wrong if they relied too much on the order of operations.

Rounding modes

Are rounding modes preserved?

- by the compiler
- by the BLAS:
  - undocumented for classical BLAS,
  - false for fast methods such as Strassen's matrix multiplication,
  - impossible for extended BLAS that are based on error free transforms (cf. TwoSum in Jean-Michel's talk).

HPC issues

Specific issues:

- order of operations: no specified order in parallel evaluations
- computing precision: problem on distributed, heterogeneous environments (not – yet – our problem)
- rounding modes:
  - is the rounding mode local to each thread or global?
  - is the rounding mode respected by the thread or set to a default value?
  - are rounding modes saved and restored at context switches during a multithreaded computation?

HPC issues

CNR: numerical reproducibility by MKL version 11: if the processors, the OS, the number of threads and the memory alignment are preserved, then MKL guarantees numerical reproducibility.
Non-efficient, non-user-friendly, non-portable solution.

Interval guarantee

New light on numerical reproducibility:

- reproducibility and correct rounding are separate notions
- a hierarchy of reproducibility levels exists: accuracy vs execution time.

Interval equivalent of the numerical reproducibility?

- the inclusion property (the guarantee that the computed result contains the exact result) must be preserved,
- preserved inclusion property and correct rounding of the exact result are separate notions,
- to guarantee the inclusion property, brute-force bounds on roundoffs errors can be used,
- a hierarchy of guarantee levels exists: accuracy vs execution time.

Interval guarantee: what to take home

Main strength of interval computations

- results may differ...

Interval guarantee: what to take home

Main strength of interval computations

- results may differ...
- ...but they are consistent:

Interval guarantee: what to take home

Main strength of interval computations

- results may differ. . .
- . . . but they are consistent:
- thank to the inclusion property, the exact result is in the intersection of all computed results.

Agenda

Existing software and libraries

- IntLab in MatLab
- intPak in Maple: not guaranteed
- IntLib in Fortran: global optimization
- COSY: Taylor models
- Boost
- MPFI
- C-XSC
- many specialized libraries, ongoing work for porting to HPC (GPU, MPI)

Agenda

References on interval arithmetic

- R. Moore: *Interval Analysis*, Prentice Hall, Englewood Cliffs, 1966.
- A. Neumaier: *Interval methods for systems of equations*, CUP, 1990.
- R. Moore, R.B. Kearfott, M.J. Cloud: *Introduction to interval analysis*, SIAM, 2009.
- W. Tucker: *Validated Numerics: A Short Introduction to Rigorous Computations*, Princeton University Press, 2011.
- S.M. Rump: *Computer-assisted proofs and Self-Validating Methods*, pp. 195-240. Handbook on Accuracy and Reliability in Scientific Computation (B. Einarsson ed.), SIAM, 2005.
- S.M. Rump: *Verification methods: Rigorous results using floating-point arithmetic*, Acta Numerica, vol. 19, pp. 287-449, 2010.

References on interval arithmetic

- J. Rohn: *A Handbook of Results on Interval Linear Problems*, http://www.cs.cas.cz/rohn/handbook 2006.
- E. Hansen and W. Walster: *Global optimization using interval analysis*, MIT Press, 2004.
- R.B. Kearfott: *Rigorous global search: continuous problems*, Kluwer, 1996.
- V. Kreinovich, A. Lakeyev, J. Rohn, P. Kahl: *Computational Complexity and Feasibility of Data Processing and Interval Computations*, Dordrecht, 1997.
- L.H. Figueiredo, J. Stolfi: *Affine arithmetic* http://www.ic.unicamp.br/~stolfi/EXPORT/projects/affine-arith/.
- *Taylor models arith.:* M. Berz and K. Makino, N. Nedialkov, M. Neher.