Exact Real Arithmetic
a (small) comparison

Norbert Th. Müller

Abteilung Informatik — Universität Trier

D-54286 Trier, Germany

non-real arithmetic - testing equality - interval arithmetic

exact arithmetic: CRCalc / XR / IC Reals / iRRAM

iRRAM internals

Background:

— *Theoretical Computer Science*

— **Countable sets:**

theory of computability and complexity well-understood

— **Non-countable sets: concurring/competing models:**

domains (Blanck, Edalat, Escardo, *Lester*(?), Tucker, ...)

$$\Leftrightarrow$$

TTE (Weihrauch, Brattka, Hertling, Ko, *Lester*(?), *M.*, ...)

$$\not\Leftrightarrow$$

BSS-model of Real RAM (Blum, Shub, Smale, ...)

Computing in accordance with TTE:

functional or imperative(!) programming

atomic objects $x \in \mathbb{R}$

fully(!) consistent with real calculus

Non-Real Arithmetic:

Fixed size arithmetic:

32bit/64bit-int: canonical semantics (mod 2^{32}, 2^{64})

Hardware

float, double: standardized semantics, IEEE 754/854

Hardware

Variable size arithmetic, e.g.:

Integer / Rational: canonical semantics

GMP (`C`, Granlund, et.al)

Multiple Precision Floats, e.g.:

MP (`Fortran 77`, R.P.Brent)

GMP (`C`, Granlund, et al.)

MPFR (`C`, Zimmermann, Lefèvre, et al.)

(variable mantissa, 32-bit-exponent, following IEEE 754/854)

until here $\uparrow$: equality $x \stackrel{?}{=} y$ easily decidable!

Algebraic numbers: canonical semantics

Leda (Mehlhorn, Näher,...)

Core library (Yap,...)

until here $\uparrow$: equality $x \stackrel{?}{=} y$ decidable (in principle) using computation diagrams + separation bounds, but ...

full real arithmetic: equality $x \stackrel{?}{=} y$ undecidable!

Searching the border of decidability:

Uniformity Conjecture (Richardson/Langley):

Consider expressions E built as follows:

(1) all (decimal) integers are allowed

(2) expressions may be built iteratively using

$$A + B \quad A - B \quad A * B \quad A/B$$

$$(A) \qquad -A \qquad e^{A} \qquad \ln A \qquad \sqrt[n]{A}$$

Conjecture: If $val(E') \leq 1$ for any subexpression $e^{E'}$, then

either $val(E) = 0$ **or** $|val(E)| > 10^{-1.3 \cdot len(E)}$

Example: logistic function (Kulisch)

$$x_{i+1} = 3.75 \cdot x_i \cdot (1 - x_i) \quad , \quad x_0 = 1/2$$

i.e.:

```
x_0=+.5000000000...   x_1=+.9375000000...   x_2=+.2197265625...
x_3=+.6429255008...   x_4=+.8608961295...   x_5=+.4490774389...
x_6=+.9277758478...   x_7=+.2512793398...   x_8=+.7055176245...
```

For UC using $3.75 = 15/4$:

$$len(E_{i+1}) = 2 \cdot len(E_i) + 8$$

so testing $x_{25} - x_{25} \stackrel{?}{=} 0$ gives E with

$$len(E) \approx 1,476,395,000$$

Interval arithmetic with variable size, eg.:

MPFI (Revol, Rouillier)

multiple precision interval arithmetic library

C software based on MPFR

look and feel: interval arithmetic...

(almost) exact arithmetic:

precise computation software (Aberth)

'range' arithmetic based on own floating point software

C++ software (no gcc 3.x...)

extended with calculus on elementary functions

($+$, $-$, $*$, $/$, x^y, sin, cos, tan, asin, acos, atan, e^x, ln$x$, max, min)

prepared for user-driven iterations:

look: exact arithmetic, feel: interval arithmetic

Example: Logistic function with ‘range’ arithmetic, Aberth

```
#include "real.h"
int main(){
  cin >> param;

  cin >> precision;  set_precision(precision);

  real x=1/real(2);  real c=375/real(100);

  for (i=1; i<=param; i++) {
    x=c*x*(one-x);
    if (i%10==0)  cout<<i<<" "<<x.str(2,20)<<endl;
  }
}
```

time for x_{1000}: 0.5 sec (using precision 950)

packages for look and feel of exact real arithmetic:

- **CRCalc (Constructive Reals Calculator, Boehm)**
- **XR (eXact Real arithmetic, Briggs)**
- **IC Reals (Imperial College Reals, Errington et al.)**
- **‘Manchester Reals’ (*David Lester*)**
- **iRRAM (iterative Real RAM, *M.*)**
- **... (many more)...**

Basic concepts of exact real arithmetic:

— Real numbers are atomic objects

— Arithmetic is able to deal with arbitray real numbers ...

— ... but usual entrace to $\mathbb{R}$ is $\mathbb{Q}$

— Underlying theory: TTE, Type-2-Theory of Effectivity ...

— ... implying: computable functions are continuous!

— ... implying: failing tests $x \leq y$, $x \geq y$, $x = y$ in case of $x = y$!

— ... using multi-valued functions

Basic methods in exact real arithmetic:

(i) explicit computation diagrams

lazy evaluation,
usually top-down-evaluation,
often using concepts from functional languages

(ii) implicit computation diagrams

reconstructable in iterations,
usually bottom-up-evaluation,
‘decision history’, ‘multi-value-cache’

CRCalc (Constructive Reals Calculator, Boehm)

- **JAVA implementation of constructive reals**
- **explicit computation diagrams with OO methods**
- **top-down evaluation based on scaled `BigInteger`**

sample program:

```
CR one = CR.valueOf(1);
CR C = CR.valueOf(375).divide(CR.valueOf(100));
CR X = one.divide(CR.valueOf(2));
for (int i=1;i<param;i++){
   X= C.multiply(X).multiply(one.subtract(X));
   if (i%10==0) {
      System.out.print(i); System.out.print(" ");
      System.out.println(X.toString(20));
   }
}
```

time for x_{1000}: 1359 sec

XR (eXact Real arithmetic, Briggs)

$x \sim \lambda i.a_i$ **for** $x = \lim_{i \to \infty} a_i \cdot 2^{-i}, a_i \in \mathbb{Z}$

python or C++ (with functional extension FC++):

```
typedef Fun1<int,Z> lambda
```

sample program:

```
int i;
XR c=QQ(375,100),x=QQ(1,2);

cout<<setprecision(20);

for (i=0; i<=param; i++) {
  x=c*x*(1-x);
  if (i%10==0) cout<<i<<" "<<x<<endl;
}
```

time for x_{1000}: 423 sec

IC Reals (Imperial College: Errington, Krznaric, Heckmann et al.)

language: `C`

linear fractional transformations using GMP big integer

Generalization of continued fractions:

$$x_{i+1} = \frac{a_i x_i + b_i}{c_i x_i + d_i}$$

lazy evaluation, top-down

lazy boolean predicates (multi-valued):

`y=realIF(2, x < 1 , a , x > -1, b)` **for**

$$y := \begin{cases} a, & \text{if } x < 1 \\ b, & \text{if } x > -1 \end{cases}$$

time for x_{1000}: 1600 sec

iRRAM (iterative Real RAM, *M.*)

language: `C++`

iterative concept (implying bottom-up evaluation)

simplified intervals

backends: GMPF/MPFR/LRGMP

limits, explicit multivalued functions, lazy boolean predicates

under construction: (restricted) lazyness

time for x_{1000}: <0.1 sec

time for x_{10000}: 17 sec, (cf. Aberth: 1468 sec, precision 17500)

More realistic example: Compute part of the harmonic series

$$h(n) = \sum_{i=1}^{n} \frac{1}{i}$$

precisely: print 10 decimals of $h(n)$!

$$\Rightarrow$$

deeply nested operations

but: not(!) hard concerning precision

computation time for $h(n)$:

	n=5,000	n=1,000,000	n=10,000,000
CRCalc	325 sec		
XR2.0	2.48 sec		
IC-Reals	0.85 sec		
Aberth	<0.1 sec	71 sec	
iRRAM	<0.1 sec	2 sec	18 sec

comparison with interval arithmetic based on hardware floats:

iRRAM:	$\approx$ **1 MFlops**	**(P3-1200)**	$\Big\}$ $\rightarrow$ **factor** $\approx 5 - 10$
`filib++`:	$\approx$ **8-22 MFlops**	**(P4-2000)**	

Example: Logistic function

```
REAL x = 0.5;  REAL c = 3.75;

for ( int i=1; i<=param; i++ ) {

  x= c*x*(1-x);

  if ( (i<100) || (i%10)==0 ) {
    rwrite(x,18); rprintf(" %d\n",i);
  }
}
```

Problem: Precision not known in advance!

Simple(?) Solution:

Repeat the computation with improved precision!

Technical Problems: How to...

...repeat?

...improve precision?

...improve computation time?

...implement intervals efficiently?

Theoretical Problems:

Which operations?

Which operators?

Discontinuous functions?!

internals of the iRRAM (iterative Real RAM)

basic idea:

iterate[1] finite approximations[2], with increasing precision

features:

ordinary arithmetic, elementary functions,
basic matrix arithmetic, complex numbers
limits[3], lazy booleans, multi-valued functions
export[4] to GMP/MPFR/`double`,
...

missing:

extended numerical part of Aberth's package...

Iterating a Computation

Sequence of precision bounds, for each iteration:

$$\overline{p}_0 \gg \overline{p}_1 \gg \overline{p}_2 \cdots$$

$\overline{p}_{i+1}$ **instead of** $\overline{p}_i$ **usually leads to smaller intervals.**

Implemented version:

$$\overline{p}_{i+1} = \overline{p}_i + \alpha \cdot \beta^i$$

with $\overline{p}_0 = \alpha = -50$ **and** $\beta = 1.25$

$\overline{p}_0$	$\overline{p}_1$	$\overline{p}_2$	$\overline{p}_3$	..	$\overline{p}_{10}$	$\overline{p}_{15}$	$\overline{p}_{20}$	$\overline{p}_{25}$	..
-50	-100	-162	-240	..	-1703	-5502	-17096	-52477	..

Heuristic: skip iterations, if 'right' precision can be estimated.

Iterations happen e.g. in case of:

- **divisions,**
  if the divding interval contains zero
- **comparisons,**
  if the compared intervals have non-empty intersection
- **printing,**
  if the interval is too large for the intended precision of output

In consequence, inifinite loops occur in case of:

- **division by zero**
- **comparison of equal numbers**

implementation: C++-exceptions

Simplified Interval Representation

Internal representation of `REAL x` as $(d - e, d + e)$

– **MP number d**

– **error information e using two longs z, p in form $z \cdot 2^p$**

$\Longrightarrow$ **simplified intervals**
$\Longrightarrow$ **acceptable error propagation, constant overhead**

Precision of arithmetic operations:
automatic, depending on e

Computation of Limits

implementation as operator with functional parameter:

```
REAL  limit   (REAL a(long,...), ....);

REAL  limit_lip   (REAL a(long,...), l,....);
```

(simplified) examples:

```
// Compute  e=2.71...
REAL e_approx (long p) {...};
REAL e()
  {
  return limit(e_approx);
  };
```

```
// Approximate sqrt using e.g. Heron‘s method
REAL sqrt_approx (long p,const REAL& x) {...};

REAL sqrt(const REAL& x)
  {
  return limit(sqrt_approx,x);
  }

// Approximate sin using Taylor series
REAL sin_approx (long p,const REAL& x) {...};

REAL sin(const REAL& x)
  {
  return limit_lip(sqrt_approx,0,x);
  }
```

usage models of the iRRAM

— full program under control of the iRRAM iterations:

e.g. computation of functions

$$f : \mathbb{R} \longrightarrow \mathbb{R}$$

— integrated into ordinary arithmetic on objects $D \subseteq \mathbb{R}$ as a function

$$\overline{f} : D \times \mathbb{Z} \longrightarrow D$$

with

$$|\overline{f}(d,p) - f(d)| \leq 2^p$$

$\Rightarrow$ extension of ordinary arithmetic with exact arithmetic through micro-iterations

e.g. reference arithmetic for `double`

e.g. defining additional functions for GMP/MPFR

iRRAM as a backend for MPFR

	10 decimals		10000 decimals	
	native	iRRAM-based	native	iRRAM-based
e(x)	0.0117 ms	0.0577 ms	201 ms	1080 ms
ln(x)	0.0388 ms	0.0696 ms	105 ms	109 ms
sin(x)	0.0427 ms	0.0745 ms	403 ms	187 ms
cos(x)	0.0270 ms	0.0678 ms	282 ms	184 ms

Installation:

- **Licence: LGPL**
- **Neccessary: GMP 4.1 / MPFR 2.x / Linux**
- **URL: `http://www.informatik.uni-trier/iRRAM/`**
- **or google: iRRAM**
- **first: `configure -with-installed=<gmp-path>`**
- **then: `make`**

Future Work

- **porting to other systems than `Linux/g++/i386`**
- **improved functions (AGM for `exp`,`sin`,...)**
- **additional functions (for `DYADIC` / `COMPLEX`)**
- **additional MP packages**
- **additional operators (for iteration)**
- **additional concepts**
  (algebraic numbers, automated differentiation...?)
- **...**
- **documentation, documentation, documentation, documentation...**

Timing examples for matrix arithmetic:

n	50	100	150	200	250	500
inversion of well conditioned matrix $H_n + 1_n$ of size $n \times n$						
Prec.	-100	-100	-100	-100	-100	-162
Memory	0.5 MB	2.1 MB	4.7 MB	8.4 MB	13 MB	52MB
Time	0.7 s	5.4 s	19 s	45s	91 s	1237 s
inversion of Hilbert matrix H_n of size $n \times n$						
Prec.	-1037	-2745	-4372	-5502	-6915	?
Memory	1.46MB	9.2MB	40 MB	81MB	152 MB	(>1GB)
Time	3.2 s	79 s	457 s	1200 s	3052 s	?

comparison to Octave:

$H_n + 1_n$	18.8 s (factor 65) for $n = 500$
H_n	impossible for $n \geq 12$

References

[Ba88] D.H. Bailey, The computation of π to 29,360,000 Decimal Digits Using Borweins Quartically Convergent Algorithm *Mathematics of Computation* **Vol. 50 Number 181 (1988) 238-296**

[Bo87] J.M. Borwein & P.B. Borwein, Pi and the AGM, A study in analythic number theory, Wiley, New York, 1987

[BoCa90] H. Boehm & R. Cartwright, Exact Real Arithmetic: Formulating real numbers as functions. In T. D., editor, *Research Topics in Functional Programming,* **43-64 (Addison-Wesley, 1990)**

[BSS89] L. Blum & M. Shub & S. Smale, On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines, *Bulletin of the AMS* 21, 1, July 1989

[Bt75] R.P. Brent, The complexity of multiple precision arithmetic, Proc. Seminar on Complexity of Computational Problem Solving, Queensland U. Press, Brisbane, Australia (1975) 126-165

[Bt76] R.P. Brent, Fast multiple precision evaluation of elementary functions, *J. ACM* **23 (1976) 242-251**

[Bt78] R.P. Brent, A Fortran multiple precision package, *ACM Trans. Math. Software* **4 (1978), pp 57-70**

[Br96] V. Brattka, Recursive characterisation of computable real-valued functions and relations, Theoret. Comput. Sci. 162 (1996),47-77

[BrHe95] V. Brattka & P. Hertling, Feasible Real Random Access Machines, *Informatik Berichte 193 - 12/1995, FernUniversität Hagen*,

[BrHe94] V. Brattka & P. Hertling, Continuity and Computability of Relations, *Informatik Berichte 164 - 9/1994, FernUniversität Hagen*,

[Br99] V. Brattka, Recursive and Computable Operations over Topological Structures, Thesis, *Informatik Berichte 255 - 7/1999, FernUniversität Hagen*

[CoAa89] S.A. Cook & S.O. Aanderaa, On the minimum computation time of functions, *Trans. Amer. Math. Soc.* **142 (1969) 291-314**

[EdHe02] A. Edalat & R. Heckmann, Computing with real numbers: (i) LFT approach to real computation, (ii) Domain-theoretic model of computational geometry. *In Gilles Barthe, Peter Dybjer, Luis Pinto, and Joao Saraiva, editors, LNCS, Springer* **2002.**

[EdPo97] A. Edalat & P. Potts, A new representation for exact real numbers, *Proc. of Mathematical Foundations of Programming Semantics 13*, **Electronic notes in Theoretical Computer Science 6, Elsevier Science B.V., 1997, URL:** `www.elsevier.nl/locate/entcs/volume6.html`

[FiSt74] M.J. Fischer & L.J. Stockmeyer, Fast on-line integer multiplication, *J. Comput. System Scis.* **9 (1974) 317-331**

[Gr00] T. Granlund, GMP 3.1, `http://www.swox.com/gmp/`

[GL00] P. Gowland, D. Lester, The Correctness of an Implementation of Exact Arithmetic, *4th Conference on Real Numbers and Computers, 2000, Dagstuhl, 125-140*

[HN00] S. Heinrich & E. Novak et al., The Inverse of the Star-Discrepancy depends linearly on the Dimension, *Acta Arithmetica, to appear*

[Kl93] R. Klatte & U. Kulisch et al., C-XSC , a C++ Class Library for Extended Scientific Computing (Springer, Berlin 1993)

[Ko91] K. Ko, Complexity Theory of Real Functions, (Birkhäuser, Boston 1991)

[Ku96] U. Kulisch, Memorandum über Computer, Arithmetik und Numerik (Universität Karlsruhe, Institut für angewandte Mathematik)

[Mm96] V. Ménissier-Morain, Arbitrary precision real arithmetic: design and algorithms, *J. Symbolic Computation, 1996,* 11

[Mu88] Müller, N.Th., Untersuchungen zur Komplexität reeller Funktionen, *Dissertation* **(FernUniversität Hagen, 1988)**

[Mu93] N.Th. Müller, Polynomial Time Computation of Taylor Series, *Proc. 22 JAIIO - PANEL '93, Part 2,* *Buenos Aires, 1993,* **259-281 (also available at** `http://www.informatik.uni-trier.de/~mueller`**)**

[Mu96] N.Th. Müller, Towards a real Real RAM: a Prototype using C++, (preliminary version), *Second Workshop on Constructivity and Complexity in Analysis, Forschungsbericht Mathematik-Informatik, Universität Trier 96-44*, **Seiten 59-66 (1996) (also available at** `http://www.informatik.uni-trier.de/~mueller`**)**

[Mu97] N.Th. Müller, Towards a real RealRAM: a Prototype using C++, *Proc. 6th International Conference on Numerical Analysis*, *Plovdiv, 1997*

[Mu98] N.Th. Müller, Implementing limits in an interactive RealRAM, *3rd Conference on Real Numbers and Computers, 1998, Paris,* **13-26**

[Mu01] N.Th. Müller, The iRRAM: Exact Arithmetic in C++, *Proceedings of CCA2000*, *Lecture notes in computer science 2064* **(Springer, Berlin, 2001) 222-252**

[Ri01] D. Richardson, The Uniformity Conjecture, *Proceedings of CCA2000*, *Springer Lecture Notes in Computer Science 2064* `http://www.bath.ac.uk/~masdr/unif.dvi`

[Rio94] M. Riordan, `ftp://ripem.msu.edu/pub/bignum/BIGNUMS.TXT`

[Sch90] A. Schönhage, Numerik analytischer Funktionen und Komplexität, *Jber. d. Dt. Math.-Verein.* 92 (1990) 1-20

[TZ99] J.V. Tucker, J.I. Zucker, Computation by 'While' programs on topological partial algebras, *Theoretical Computer Science* 219 (1999) 379-420

[We87] K. Weihrauch, Computability (volume 9 of: *EATCS Monographs on Theoretical Computer Science*), (Springer, Berlin, 1987)

[We95] K. Weihrauch, A Simple Introduction to Computable Analysis, *Informatik Berichte 171 - 2/1995, FernUniversität Hagen*

[We97] K. Weihrauch, A Foundation for Computable Analysis, *Proc. DMTCS´96*, **(Springer, Singapore, 1997) 66-89**

[Zi00] P. Zimmermann, MPFR: A Library for Multiprecision Floating-Point Arithmetic with Exact Rounding, *4th Conference on Real Numbers and Computers, 2000, Dagstuhl, 89-90*, see also `http://www.loria.fr/projets/mpfr/`