OI-OPPA. European Social Fund
Prague & EU: We invest in your future.

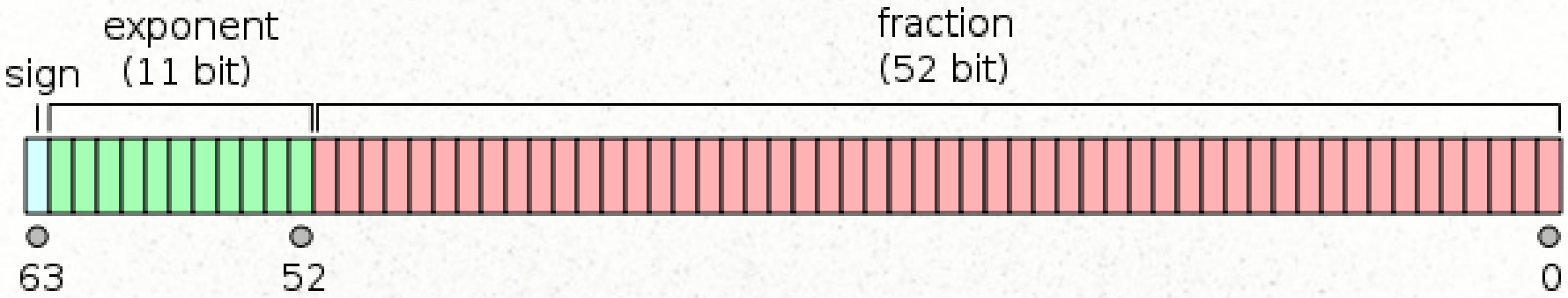
CGAL number types

Jaroslav Minařík

3.14159265358979323846264338327950288419716939937510582097494459

Double

Double-precision binary floating-point format



$$= (-1)^{\mathrm{sign}}(1.b_{-1}b_{-2}...b_{-52})_2 \times 2^{e-1023}$$

[3]

Number types, number types...

int
short int
long int
long long int
float,
double
long double
CORE::BigInt
CORE::BigRat
CORE::BigFloat
CORE::Expr
leda_integer
leda_rational
leda_bigfloat
leda_real

mpz_class
mpq_class
Gmpz
Gmpq
Gmpzf
Gmpfr
Gmpfi
MP_Float
Interval_nt
Lazy_exact_nt
Quotient

Built-in number types

- Float, double, long double
- Basic arithmetic and comparison operators
- CGAL::is_finite(x)

CGAL number types

MP_Float

- Multiple-precision
- + - * computed exactly
- Mantissa limited only by memory
- Can be initialized from std::stream
- Quadratic complexity for multiplications

approximate_division(a,b)

approximate_sqrt(a)

CGAL number types

Lazy_exact_nt<NT>

- Approximates computation
- Returns exact output only if required
- Can be initialized from std::stream

CGAL number types

Lazy_exact_nt<NT>

set_relative_precision_of_to_double(d)

get_relative_precision_of_to_double()

exact()

approx() - returns interval containing the exact value

CGAL::Lazy_exact_nt<CGAL::Gmpfr>

CGAL number types

Sqrt_extension<NT,ROOT>

- Square root extension of the form

  a0+a1*sqrt(root)

```
CGAL::Sqrt_extension<int,int> extension(1,2,3);
// represents 1+2*sqrt(3)
```

CGAL number types

Quotient<NT>

q.numerator() - returns a numerator of q.

q.denominator() returns a denominator of q.

to_double(q) – approx. to q

is_valid(q)

is_finite(q)

sqrt(q)

CGAL::Quotient<CGAL::MP_Float>

GMP number types

Gmpz

- Arbitrary precision integer
- Supports bit shifts and bitwise operators

Gmpq

- Rational number
- Initializable with numerator and denum.

GMP number types

Gmpzf

- Multiple-precision floating-point type
- $m*2^e$

  m – arbitrary precision integer

  e – long
- Operations + - * and integral_division()

GMP number types

Gmpfr

- Fixed precision floating-point type
- Uses rounding modes (nearest, zero, down, up)

get/set_default_precision()
get/set_default_rndmode()

add(a,b)
f.abs(Precision_type p)
f.sqrt(Precision_type p)
f.kthroot(int k, Precision_type p)
f.square(Precision_type p)

GMP number types

Creation

CGAL::Gmpzf f(long int i); // creates Gmpzf initialized with i

*CGAL::Gmpfr f(std::pair<CGAL::Gmpz,long> ie); // ie.first * $2^{ie.second}$*

CGAL::Gmpq q (unsigned long n, unsigned long d); // fraction n/d

std::istream& in >> &q; // reads a number from in, then converts it to a Gmpq

```
typedef CGAL::Lazy_exact_nt<CGAL::Quotient<CGAL::MP_Float> > NT;
typedef CGAL::Cartesian<NT> K;
```

LEDA number types

leda_integer

- Integer of arbitrary length
- Exact computation in $\mathbb{Z}$
- Limited by resources of computer

leda_rational

- Exact computation in $\mathbb{R}$
- Limited by resources of computer

LEDA number types

leda_bigfloat

- Variable precision floating-point type
- Rounding mode and precision (mantissa length) can be set

leda_real

- Operations + - * / k-th root

CORE number types

BigInt

- Exact in $\mathbb{Z}$

BigRat

- Exact in $\mathbb{R}$

BigFloat

- Variable precision floating-point type
- Rounding mode and precision can be set

Interval arithmetic

Interval_nt<Protected>

- Represents intervals (double as endpoints)
- boolean Protected

Gmpfi

- Represents intervals with Gmpfr endpoints

Interval arithmetic - interface

- to_double() - middle of interval, approx.
- inf() - lower bound
- sup() - upper bound
- is_point() - true whether bounds are equal
- is_same(interval) – compares bounds
- do_overlap(interval)

Interval arithmetic - usages

```
CGAL::Interval_nt<true> a ( double left, double right); // interval [left; right]

CGAL::Gmpfi b (
      CGAL::Gmpfr left,
      CGAL::Gmpfr right,
      CGAL::get_default_precision()
); // interval [left; right]
```

CGAL::Gmpfi c;
std::istream &is >> c; //Reads c from is. is must have the form [inf,sup], where inf and sup have valid Gmpfr input formats.

c.sqrt(Precision_type p) // Returns the square root of i, with precision p. If p is not specified, the precision used is the maximum between i's precision and the default.

Thank you for your attention

Sources

- [1] CGAL User and Reference Manual 3.9
- [2] http://www.cgal.org/Manual/latest
- [3] http://en.wikipedia.org/wiki/Double_precision (image)
- [4] http://hacksoflife.blogspot.cz/2009/04/why-cgal.html

OI-OPPA. European Social Fund
Prague & EU: We invest in your future.