The design of CoCoALib/CoCoA

Anna M. Bigatti – Università di Genova, Italy

June 2023 – EMS Lyon

What is CoCoA?

History: CoCoA-1 (1989)

Giovini-Niesi-Robbiano **Aim**: a *mathematician-friendly* software for
Computations in **Co**mmutative **A**lgebra
especially **Gröbner bases**. Only on Macintoshes. Written in Pascal.

Second life: CoCoA-3/4 (1995)

Capani-Niesi/Abbott-Bigatti, Robbiano
With dedicated CoCoA Language. All platforms. Written in C.

Third life: CoCoA-5 (2010-) and CoCoALib (2003-)

Abbott-Bigatti-Robbiano

- open source GPL **C++ software library**
- **interactive system** (*Demo*)
- prototype OpenMath-based **server**

Evolved CoCoA Language. Designed to be a C++ library.

What is CoCoA?

History: CoCoA-1 (1989)

Giovini-Niesi-Robbiano **Aim**: a *mathematician-friendly* software for
Computations in **Co**mmutative **A**lgebra
especially **Gröbner bases**. Only on Macintoshes. Written in Pascal.

Second life: CoCoA-3/4 (1995)

Capani-Niesi/Abbott-Bigatti, Robbiano
With dedicated CoCoA Language. All platforms. Written in C.

Third life: CoCoA-5 (2010-) and CoCoALib (2003-)

Abbott-Bigatti-Robbiano

- open source GPL **C++ software library**
- **interactive system** (*Demo*)
- prototype OpenMath-based **server**

Evolved CoCoA Language. Designed to be a C++ library.

CoCoA-5 main functions and operations

What can I compute with **CoCoA**?

- **Gröbner bases of ideals/modules**, wide choice of term orderings
- special handling for **ideals of points** and **monomial ideals**
- **Hilbert series**, *resolutions*, *Betti numbers*
- polynomial **factorization**
- basic **exact linear algebra** (`LinSolve`, `LinKer`, `eigenvectors`, `det`)
- **approximate points**: border bases, polynomial relations
- **real roots** of univariate polynomials
- **0-dimensional ideals** (radical, IsMaximal, MinPoly, PrimaryDecomposition..)
- **Implicitization of Hypersurfaces**
- **SAGBI bases**
- .....

More than **CoCoA** itself

Several ways of extending **CoCoA-5**

- Write your own functions in **CoCoA-5** language

```
define StrangeFunction(X)
  if   type(X) = INT then return 2^X;
  elif type(X) = MAT then return det(X);
  endif;
  return X;
enddefine;
```

- Collect some functions into a new **CoCoA-5 package**

- Write the new functions in C++ inside **CoCoALib**,
  and then make them "visible" to **CoCoA-5**
  (the new interpreter makes this last step really easy!)

From **CoCoA-5** to **CoCoALib**: example 1

Quick implementation of prototype algorithms in **CoCoA-5**, then translate it into C++ for better performance within **CoCoALib**

To facilitate this process in **CoCoALib** we use:

- same function names as in **CoCoA-5** (whenever possible)
- *functional syntax e.g.* `deg(f)`
- object-oriented *method call* `f.myDeg()` only for extreme efficiency

```
Use QQ[x,y,z];
I := ideal(x^3 + x*y^2 - 2*z, ...., ...., .... );
GBasis(I);    // same as  "print GBasis(I);"
```

This small sample literally translates into:

```
ring P = NewPolyRing(RingQQ(), symbols("x,y,z"));
ideal I = ideal(RingElems(P, "x^3 + x*y^2 - 2*z, ........ ");
cout << GBasis(I);
```

From **CoCoA-5** to **CoCoALib**: example 2

```
use R ::= QQ[a];
K2  := NewQuotientRing(R, "a^2-2");   // K2 is QQ[a]/(a^2-2)
psi := CanonicalHom(R, K2);           // psi: QQ[a] --> QQ[a]/(a^2-2)

use R; // polynomials are read as elements in R = QQ[a]
f := 1/psi(a^2 + 2*a -1);             // gives  ((2/7)*a -1/7) in K2
f;
```

This sample literally translates into:

```
ring R  = NewPolyRing(RingQQ(), symbols("a"));
ring K2 = NewQuotientRing(R, "a^2-2");
RingHom psi = CanonicalHom(R, K2);

RingElem f = 1/psi(RingElem(R, "a^2 + 2*a -1"));
cout << "f is " << f << endl;
```

From CoCoALib to built-in function in CoCoA-5

Design goal of the CoCoA-5 intepreter

easy to expose CoCoALib functions to CoCoA-5

Best example: "One-liner"

The function `JanetBasis` expects an ideal
(and outputs a list of polynomials)

The code to expose it to CoCoA-5 is just **one line** (*C macro*)

```
DECLARE_COCOALIB_FUNCTION1(JanetBasis, IDEAL)
```

meaning: 1 argument of type `IDEAL` (wrapper for CoCoALib `ideal`)

Output type is automatically determined and wrapped up for **CoCoA-5**

CoCoALib: the C++ mathematical brain of **CoCoA-5**

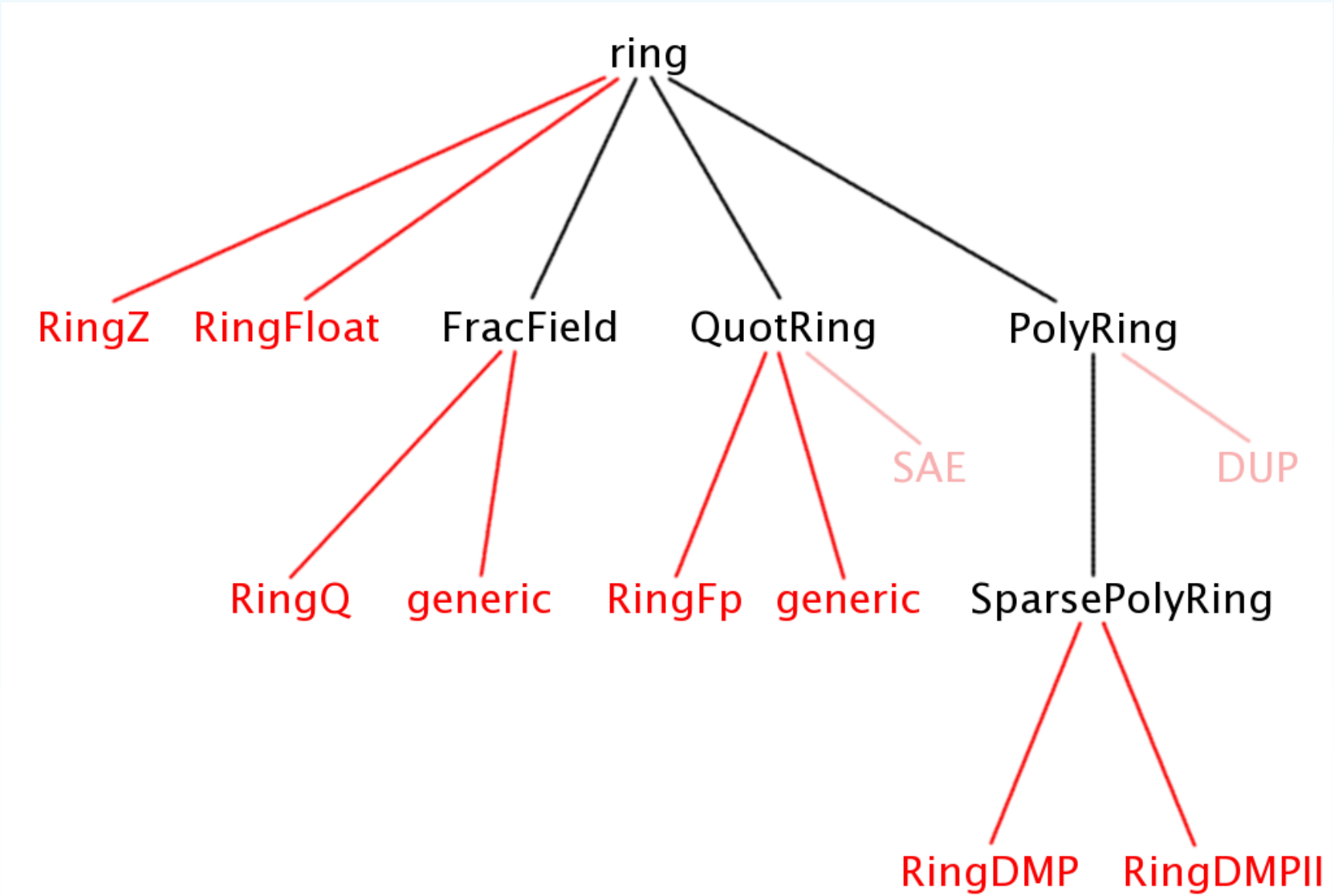
- (aim) all **CoCoA-5** functionalities available in **CoCoALib**
- **CoCoA-5** interpreter: easy to *expose* **CoCoALib** functions

CoCoALib: C++ library

- Designed to be **easy to use**
- Execution speed is **good**
- Well-documented, including **many examples programs**

- **Free and open source** C++ code (GPL3 licence)
- Source code is **clean and portable** (C++14)
- Design respects the underlying **mathematical** structures (inheritance, no templates)
- **Robust** (Motto: “No nasty surprises”), exception-safe, thread-safe

Ring Inheritance Diagram



Contributions

Code authors

CoCoALib (John Abbott & Anna Bigatti)
Parser and interpreter for **CoCoA-5** (Giovanni Lagorio)

But the openness and clean design of the library
was chosen to encourage contributions

Direct contributions to **CoCoALib**

- Mathematical support and feedback (L. Robbiano)
- Gröbner bases structure, ideal/module operations (M. Caboara)
- Mayer-Vietoris trees (E. Sáenz de Cabezón)
- Janet and Pommaret Bases (M. Albert and W. Seiler)
- Approximate points (M. L. Torrente and C. Fassino)

Adding external libraries

External libraries integrated with **CoCoALib**

- B. Roune: ***Frobby*** (monomial ideals)
- C. Söger: ***Normaliz*** (affine monoids or rational cones)
- A. N. Jensen: ***GFan*** (Gröbner fans and tropical varieties)
- A. Griggio: ***MathSAT*** (Satisfiability modulo theories (SMT) solver)

Example: **libnormaliz**

One file: `AlgebraicCore/ExternalLibs-Normaliz.C` (and `.H`)

- definition of the (CoCoALib) class `cone`
- functions for data conversions between the two libraries
- functions actually available to the CoCoALib user

Utilities: `tests/test-normaliz1.C` and `examples/ex-normaliz1.C`

```
./configure --with-libnormaliz=PATH-TO/libnormaliz.a
make
```

In conclusion....

I hope there was
a little taste of **CoCoA** for everyone ;-)

Thank you!

cocoa@dima.unige.it
http://cocoa.dima.unige.it