CoCoALib

a C++ library for Computations in Commutative Algebra

John Abbott
Università di Genova, Italy

Outline

- What is CoCoALib?
  - The old and the new
  - Current state
  - Design philosophy
  - Inheritance
  - Div Masks
  - Future Plans
  - How to join in
- Some examples of CoCoALib
  - Empty
  - Change of coordinates

4, Lib, Server, and 5?

- CoCoA-4 current system 4.7.5 (old and arthritic, in C)
- CoCoALib C++ library (young, spritely and flexible, open source)
- CoCoAServer is a prototype server program; can be called from CoCoA-4, to use some features of CoCoALib. Easily extensible.
- CoCoA-5 future system whose core will be CoCoALib, with extended language and capabilities

ApCoCoALib is a C++ library built on top of CoCoALib, developed by the team in Germany (`http://www.apcocoa.org`). It extends CoCoAServer; there is also ApCoCoA which extends CoCoA.

Current state

- types for representing poly. rings, ideals and submodules, matrices
- the coefficient rings include $\mathbb{Q}$, $\mathbb{F}_p$, $\mathbb{R}$, and $k(a_0, \ldots, a_n)$
- general term-orderings and multi-gradings (for both poly. rings and modules over them)
- Gröbner bases and several other ideal/module operations (faster and more flexible than CoCoA-4)
- ring homomorphisms for mapping values between rings
- Hilbert function and factorizer (transplanted from CoCoA 4)
- Weyl Algebras advanced prototype implementation
- Easy access via prototype CoCoAServer from CoCoA-4.

We develop our code on **GNU/Linux** machines and **MacOS X**.

We use **GMP** for big integer arithmetic and high precision floats.

Design Philosophy behind CoCoALib

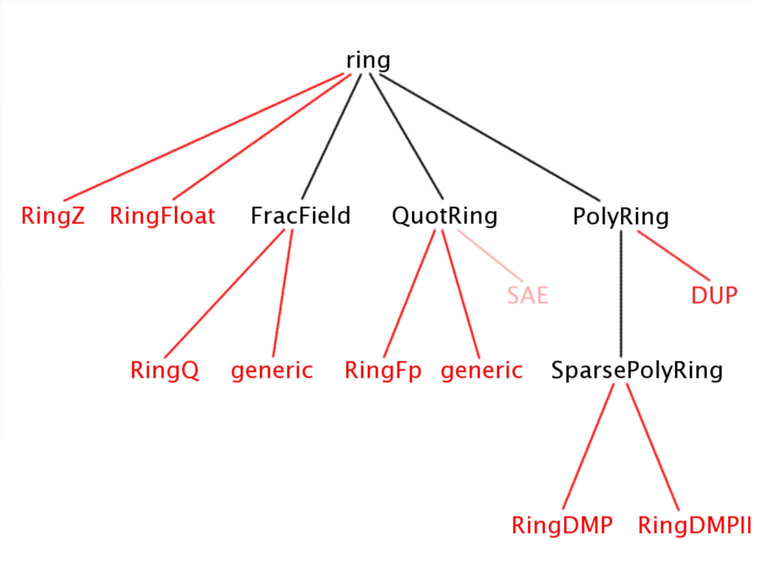
The development requires an enormous investment of time and resources. To justify this effort, CoCoALib must become popular.

Basic goals of the design to achieve popularity:

- the code must be **easy and natural** to use
- the code must exhibit **good run-time performance**
- the source code must be clear and **well designed**
- the source code must be **well documented** (users & maintainers)
- the source code must be clean and **portable**

Firm mathematical basis (in tandem with Robbiano & Kreuzer's book)

Ring Inheritance Diagram



DivMask Implementation

Idea: define map $\phi : PP \to \{0,1\}^s$ from PPs to s-bitsets s.t.

$$t | t' \quad \Longrightarrow \quad \phi(t) \subseteq \phi(t')$$

Such ϕ are **DivMask rules**; many exist, none is universally best.

Example: $s = 32$ bits

PP:	x_0^2	x_1^0	x_2^0	x_3^5	x_4^0	x_5^3	x_6^1	x_7^0	x_8^3	x_9^1	...
Bitset:	1	0	0	1	0	1	1	0	1	1	...

C++ Inheritance: user can choose DivMask rule at run-time, so computing a DivMask is “slow”, but subset test is the same for all rules $\Longrightarrow$ inline $\Longrightarrow$ fast.

Some Future Plans

- CoCoA-5, new interactive system with improved language & better errors
- develop CoCoAServer with full interpreter
- self-saturating algorithm for non-homogeneous Gröbner bases
- redesign implementation of ideals
- see CoCoALib Task Table for more details

How to join in

Prerequisites

- Some knowledge of basic C++ programming
- Mild familiarity with compilation and `make`
- the GMP library

What to do

- Download CoCoALib from http://cocoa.dima.unige.it/cocoalib/
  current version: `CoCoALib-0.9930`
- Configure and compile
  ```
  ./configure; make
  ```
- Play and experiment!
  ```
  cd examples; make
  ```