

# The CoCoA Project

The CoCoA project comprises a small group of researchers studying

*Computations in Commutative Algebra*

Alongside the usual academic publications, the group also produces the program CoCoA which is a *special-purpose* computer algebra system whose particular strengths include: Gröbner cp bases, polynomial factorization, exact linear algebra, Hilbert functions, ideals of points, and toric ideals.

The software is *freely available* for research and educational purposes: the latest official public release is CoCoA 4.6 which can be obtained from our **web site** as well as from the mirror site in USA.

Last year the completely new CoCoA 5 library became available as an alpha release. Compared to CoCoA 4.6 this new library offers enhanced performance, and will eventually offer a wider range of features.

## What is CoCoA 5?

The CoCoA program, first in series 3 and now in series 4, has enjoyed increasing popularity thanks to its user-friendly interface coupled to good performance. Indeed, it has grown far beyond what was originally foreseen ten years ago at its birth. After due consideration, we concluded that the best way to resolve some limitations inherent in the current design would be to start afresh.

CoCoA 5 is a **completely new implementation** destined to replace the current series. The design of CoCoA 5 offers many advantages over that of series 4. For instance, the issue of interoperability of programs is becoming ever more important, but the facilities of CoCoA 4 are accessible **only as an interactive system**. Instead the capabilities of CoCoA 5 are to be available in several guises:

- ◇ as a **C++ library** which is clean, swift and well documented
- ◇ as a **server** which communicates using “OpenMath”
- ◇ as a **standalone interactive system** — a UI which sends requests to the server.

## Design Goals for CoCoA 5

Our design is guided by the tenet that CoCoA 5 is to be better than CoCoA 4 in every respect (*e.g.* faster, easier to use, more capabilities). The main areas of improvement are:

- ◇ firmer mathematical basis underpinning the software
- ◇ wider choice of coefficient rings
- ◇ exploitation of multi-graded structure in Buchberger's algorithm
- ◇ easier to create and use different rings contemporaneously
- ◇ significant gain in run-time performance

We regard ease of use as being of paramount importance, and shall safeguard it in the design of CoCoA 5.

## Current state of CoCoA 5

The **library** is the most evolved component: a first version was released in 2003, and a major revision is anticipated for late summer 2004. A prototype **server** was demonstrated at ISSAC, and will be released in late 2004 with a compatible **CoCoA 4.4** to act as an interim user interface.

The facilities offered by the **library** include an efficient implementation of **Buchberger's algorithm** together with the following features:

- ◇ types for representing **polynomial rings, ideals and submodules**
- ◇ general **term-orderings** and **multi-gradings** (incl. for modules) following the book **Robbiano and Kreuzer**, volume 2
- ◇ **Gröbner bases** for ideals and modules (faster than CoCoA 4)
- ◇ the **coefficient rings** offered are rationals, modular integers, floating point numbers and rational functions of one parameter
- ◇ **ring homomorphisms** for transporting values from one ring to another

## Continuing Development

In the **immediate future** work on **CoCoA 5** will be largely directed at:

- ◇ implementing **new ring types**, *e.g.* algebraic extensions
- ◇ refining the code for **modules and matrices**
- ◇ **fine-tuning** of Buchberger's algorithm
- ◇ improvements to the **server component**
- ◇ numerical applications

The new interactive user interface component of **CoCoA 5** is still distant. In the interim, future releases in series 4 will allow access to many of the features of the **CoCoA 5 library** via the **server**.

## Implementation in C++

The **CoCoA 5 library** and **server** are implemented in C++. Judicious use of inline functions and inheritance in the **library** pays dividends in terms of **simplicity, flexibility, and speed**.

Examples of some abstract classes are: **ring, QuotientRing, PolyRing**.

Examples of some concrete classes represent:  $\mathbb{Z}$ ,  $\mathbb{F}_p$ , and  $R[x_1, \dots, x_n]$

The **library** favours **safe programming**: where possible, operations employ a natural syntax and include extensive checking. Many operations offer a **fast unsafe** alternative which can be used where efficiency is crucial.

There are **several debugging** aids in the **library**; for example, customized memory management includes debugging and statistical facilities which can be disabled when speed is important.

## CoCoA Contact Information

For software downloads, and more information about **cocoa**, visit our web site:

<http://cocoa.dima.unige.it>

You can also get downloads from the mirror in America at this URL

<ftp://ftp.reed.edu/mirrors/cocoa>

We develop our code on **GNU/Linux** and **MacOS X** machines and verify portability to other operating systems.

The designer of **CoCoALib** is John Abbott.

The managers of the CoCoAproject are

Lorenzo Robbiano and John Abbott in Genoa (Italy)  
Martin Kreuzer in Dortmund (Germany)

The main programmers are

John Abbott   Anna Bigatti   Massimo Caboara

## Example Program using CoCoA 5 Library

```
PolyRing Qx = NewPolyRing(RingQ(), 2); // default names are "x[0],x[1]"
PolyRing Qy = NewPolyRing(RingQ(), SymbolRange("y",0,5));
const vector<RingElem>& x = indets(Qx);
const vector<RingElem>& y = indets(Qy);

vector<RingElem> images;
images.push_back(3*y[0] - 7*y[1]);
images.push_back(-y[0] + y[3]);
RingHom phi = PolyAlgebraHom(Qx, Qy, images);

cout << "Ring homomorphism phi: Qx --> Qy" << endl
      << "  sends " << x[0] << "  to " << image[0] << endl
      << "  and   " << x[1] << "  to " << image[1] << endl;

RingElem f = power(x[0]+2*x[1], 3);
cout << f << "  maps to " << phi(f) << endl;
```