The CoCoA project dates back to 1987 under the lead of L. Robbiano: the aim was to create a mathematician-friendly laboratory for studying Commutative Algebra, especially Gröbner bases. Since then, always maintaining this tradition, it has evolved and been rewritten, and now offers:

- an open source C++ software library, CoCoALib
- a new interpreter for the interactive system, CoCoA-5
- a prototype OpenMath-based server

The openness and clean design of CoCoALib and CoCoA-5 are intended to offer different levels of usage, and to encourage external contributions.

### The CoCoA language

- **Mathematical syntax for lists**
  
  ```
  [ n in 1..50 | IsPrime(n) and IsPrime(n+2) ];
  ```

- **No declaration of variables (dynamically typed)**
  
  ```
  use QQ[x,y,z];
  A := (x+y)^3 + (z-2);  // type(A) is now RINGELEM
  ```

- **Flexible conversion from lines of code into a function**

  ```
  define ParseCode(X);    if type(X) = INT then return 2^X; 
                             endif;
  ```

- **ring homomorphisms**

  ```
  phi := CanonicalHom(R, R);
  psi := CanonicalHom(R, R2);
  theta := CanonicalHom(K1,P(2));
  ```

- **many constraints of the CoCoA-4 language avoided by removing default “invisble multiplication” (xy means *xy).* Still allowed inside triple-`::`

  ```
  I := *** Ideal(2*x-2*y-z, 2*x*z+2*y-z, 2*y-z) ***;
  ```

- **rings and functions can be assigned and passed as arguments**
- **improved error messages**

### New in the CoCoA-5 language

The new system (mostly compatible with CoCoA-4) provides greater expressibility and a more solid mathematical basis.

- **full flexibility for the field of coefficients: for example algebraic extensions and fraction fields of any ring, and even heuristically guaranteed floating point arithmetics with rational reconstruction**

  ```
  use R := QQ[a];
  K1 := NewFractionField(R);  // K1 is QQ(a)
  ```

- **ring homomorphisms**

  ```
  phi := CanonicalHom(R, R);
  psi := CanonicalHom(R, R2);
  theta := CanonicalHom(K1,P(2));
  ```

- **many constraints of the CoCoA-4 language avoided by removing default “invisble multiplication” (xy means *xy).* Still allowed inside triple-`::`

  ```
  I := *** Ideal(2*x-2*y-z, 2*x*z+2*y-z, 2*y-z) ***;
  ```

- **rings and functions can be assigned and passed as arguments**
- **improved error messages**

### Extending CoCoA-5

There are several ways of extending CoCoA-5:

- **Add a new function written in CoCoA-5 language**
- **Collect 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!)

---

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

- **Designed to be easy to use:**
  - free and open source C++ code (GPL3 licence)
  - well-documented C++ code (html, 200 pages pdf), many examples (in examples/)
  - function interfaces are natural for mathematicians
  - design inspired by and respects the underlying mathematical structures
  - source code is clean and portable
  - execution speed is good with robust error detection

### Main types in CoCoALib

- **BigInt, BigRat** (C++ wrappers for GMP integers and rationals)
- **rings** (ring, PolyRing, QuotientRing, FractionField, ...)
- **power-product monoids** (PPMonoid, PPMonoidEv, PPMonoidOv, ...)
- **matrices** (matrix, DenseMatrix, MatrixView)
- **homomorphisms**
- **ideals** (with optimized square-free/monomial ideals)
- **modules and submodules**

### Main functions/operations in CoCoALib

- **Gröbner bases of ideals/modules with wide choice of term orderings**
- **special handling for ideals of points, monomial ideals, etc.**
- **Hilbert series, resolutions, Betti numbers**
- **polynomial factorization**
- **basic exact linear algebra** (LinSolve, LinKer, eigenvectors, det)
- **approximate points, border bases, approximate polynomial relations**

### Compatibility of CoCoA-5 with CoCoALib

Do you have a promising prototype of your algorithm in CoCoA-5 and you want to translate it into C++ for better performance?

To facilitate the conversion we have used:

- **same function names in CoCoA-5 and CoCoALib (whenever possible)**
- **functional syntax**

  ```
  (e.g. GBasis(I) rather than object oriented method call I.GBasis())
  ```

### Direct contributions to CoCoALib

- **Mathematical support and feedback** (by L. Robbiano)
- **Gröbner bases structure and ideal/module operations** (by M. Caboara)
- **Mayer-Vietoris trees** (by E. Sáenz de Cabezón)

### 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)
- **Experimental interface with GSL** (GNU Scientific Library)

### Other contributions to CoCoA-5 and CoCoALib

- **CoCoA-5 packages** (by E. Palezzato, G. Bianco)
- **CoCoA-5/CoCoALib documentation feedback** (by A. Caleo, E. Fascio)

---

**CoCoA-5** is a system for doing Computations in Commutative Algebra

**CoCoALib** is a C++ library for doing Computations in Commutative Algebra

Available at [http://cocoa.dima.unige.it/](http://cocoa.dima.unige.it/) and [http://cocoa.dima.unige.it/cocoalib](http://cocoa.dima.unige.it/cocoalib)