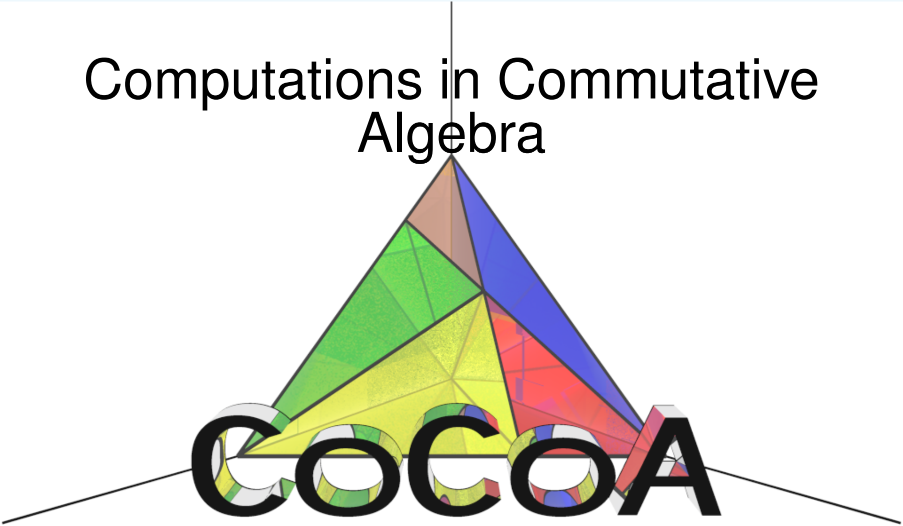
Computations in Commutative Algebra



John Abbott and Anna Maria Bigatti
Università di Genova, Italy

Outline

You already know a lot about CoCoA!

- A guided tour in the CoCoA web page
- Some little known gems of CoCoA
  - The online help
  - Making lists
  - Making matrices
  - Approximations
  - Printing

A guided tour in the CoCoA web page

What is CoCoA?

- CoCoA is a specialized computer algebra system.
- Its programming language is designed for non-programmers: no declarations, “natural mathematical syntax”, ...
- multilingual page: very simple examples of CoCoA syntax (please add more languages!)

Help system

- Reference card,
- (GUI) html help, pdf version (about 400 pages),
- Google search.

CoCoAForum and CoCoAWiki

publications, download, conferences, Kreuzer-Robbiano book, ...

The online help

Optimal use of the online manual:
within CoCoA type `?` followed by a search key (the search is case insensitive), *e.g.* this produces the manual entry for “Mat”

```
? mat
```

If no exact match for the search string is found then all keywords containing it are shown, try:

```
? ma
```

The command `??` displays **all** keywords in the online help system that match the search key:

```
?? mat
```

Making lists

The most elegant way to make a list is via the “mathematical syntax”

```
[X^3 | X In Indets()];
```

but note that you may use only **one** finite set of indices.

For example to create the list of pairs for the Buchberger algorithm, $\{(i,j) \mid 1 \leq i < j \leq 4\}$, you may use the `CartesianProduct`:

```
[ Pair In (1..4)><(1..4) | Pair[1] < Pair[2] ];
```

or the function `Flatten`:

```
L := [ [ [I,J] | J In (I+1)..4 ]  | I In 1..3 ];
Flatten(L, 1);
```

Making matrices

The standard way to define a matrix is by writing a list of lists:

```
Mat([ [1, 1, 1],  [0, 0, -1],  [0, -1, 0] ]);
```

There are other ways, some introduced for this school (CoCoA-4.7.2)!

```
MakeMatByRows(3, 3, [1, 1, 1,  0, 0, -1,  0, -1, 0]);
MakeMatByCols(3, 3, [1, 0, 0,  1, 0, -1,  1, -1, 0]);

RowMat(Indets());   ColMat(Indets());   DiagMat(Indets());

MatConcatAntiDiag(LexMat(2), DegRevLexMat(3));

BlockMatrix([ [0, LexMat(2)],  [DegRevLexMat(3), 0] ]);
...
```

See `??mat`.

Approximations

CoCoA likes exact computations, but there are some functions dealing with “approximations”.

In this school you will use parts of the Numerical package developed by the team in Dortmund. See `??numerical`.

`RealRootsApprox` computes rational approximations for the real roots of a univariate polynomial over Q.

You may use the functions `DecimalStr, FloatStr` to **visualize** a rational number as a decimal approximation:

```
RR := RealRootsApprox(x^2-20000);
RR[2];
FloatStr(RR[2]);
DecimalStr(RR[2]);
```

See also `??approx`.

Printing

The function Latex prints the LaTeX expression of your data: ready to be copied into your papers ;-)

```
Latex(DegRevLexMat(4));
```

The function StarPrint prints asterisks for all multiplications: ready to be copied as input into other programs...

```
F := x^3+2xy-y^2;
StarPrint(F);
```

The function Format makes a string with an appropriate number of blank spaces (useful for a aligned output ;-)

```
Print Format("ciao", 20);
Print Format((x-y)^2, 20);
```

See ??print.