Computer Science Journal of Moldova, vol.**4**, no.3(12), 1996

Some Features of CoCoA 3

A.Capani G.Niesi L. Robbiano

Abstract

CoCoA is a special-purpose system for doing *Computations in Commutative Algebra*. It is the ongoing product of a research team in Computer Algebra at the University of Genova (Italy), whose members are: Lorenzo Robbiano (team manager), Gianfranco Niesi, Antonio Capani (CoCoA authors), Anna Bigatti, Massimo Caboara, Gabriel De Dominicis and occasionally other researchers and students.

1 Introduction

From the very beginning CoCoA has been designed to offer maximum ease of use and flexibility to the mathematician with little or no knowledge of computers. It belongs to the class of Computer Algebra Systems which are specialized in doing computations on polynomials and which includes for instance Macaulay [BS], Macaulay 2 [GS] and Singular [GPS].

The architecture of CoCoA is designed to offer flexibility through *efficiency*, *portability*, *programmability*, and *multiple coefficient arithmetics*.

Currently CoCoA runs on the following platforms: Sun4 workstations, DEC-Alpha station, Macintosh, PC (under DOS and Linux). The system can be obtained by anonymous ftp from the URL:

`ftp://lancelot.dima.unige.it/cocoa`

1.1 The History

CoCoA is the name of a research project, which started in the mid-eighties. It was coordinated by Robbiano and its first official version was due to Alessandro Giovini and Gianfranco Niesi [GN]. The original idea was to bring together high level know-how from Computer Science and Mathematics, and produce a piece of software which could be both a useful tool for the working mathematician and a laboratory for research in different fields, ranging from Commutative Algebra to Software Engineering. After the premature death of Giovini (January 1993) a new project started. Robbiano continues to act as the project manager, Antonio Capani and Gianfranco Niesi are the authors and many other researchers and students contribute at different levels. We like to mention Anna Bigatti for the Hilbert-Poincaré Package, Massimo Caboara and Gabriel De Dominicis for the Hilbert-Driven Package. We are also pleased to thank Martin Kreuzer and Anthony Geramita for helping our team with their continual support and ideas.

1.2 The Current Use of CoCoA

Currently CoCoA is used by researchers in several countries. Most of them are Commutative Algebraists and Algebraic Geometers, but also people working in different areas like Analysis and Statistics have already benefitted from our system.

CoCoA is also used as the main system for teaching advanced courses in several Universities. Besides Italy, the most intensive use is by Tomas Recio at the University of Santander (Spain), Anthony Geramita at the Queen's University (Canada) and Martin Kreuzer at the University of Regensburg (Germany).

1.3 The Main Features

CoCoA is able to perform both simple and sophisticated operations on multivariate polynomial rings and on various data connected with them (ideals, modules, matrices, rational functions). Arbitrarily large integers are also available. Every computation is performed within a ring

(or "ring-environment"). The user can define and use many rings, but at any one time there exists a chosen *current ring*.

The system is capable of performing *basic* operations such as: sums, products, powers, derivatives, gcd, and lcm of polynomials; sums, products, powers, and derivatives of rational functions; sums, products, and powers of ideals; sums of modules; sums, products, powers, determinants, and adjoints of matrices; operations between heterogeneous values, like the product between an ideal and a polynomial, and so on.

Besides these, more advanced operations are available. For example: Gröbner bases of ideals and modules; syzygies of ideals and modules; minimal free resolutions of ideals and modules; intersection and division of ideals and modules; inclusion and equality test for ideals and modules; elimination of indeterminates; homogenization of ideals; Poincaré series and Hilbert functions.

The core of the system is a very fast implementation of algorithms for computing Gröbner bases, syzygies and Poincaré series of ideals and modules over a polynomial ring whose coefficient ring is a field. These algorithms have been optimized in several directions [GMNRT, CDNR, B, T, CDR] and are used as a foundation for many operations. Most users can, however, completely ignore the theory of Gröbner bases and even their existence: CoCoA will do all the necessary "Gröbner stuff" in the background. However, for optimum use of the system some knowledge of the theory is helpful.

CoCoA offers a Pascal-like programming language, named CoCoAL, that allows the user to customize the system and extend the embedded library.

1.4 The Architecture

CoCoA has a kernel written in the C language (about 55,000 lines) for portability and efficiency, and a library written in CoCoAL (about 1,500 lines), the high–level language of the system.

It has different interfaces for different machines; each interface interacts with the system using a Low Level Protocol (LLP) which is platform independent.

The interface-independent part of CoCoA is divided into two modules: ENGINE and MATH. The first one is the main motor: it includes the interpreter of the CoCoAL language which allows it to exchange LLP-inputs and LLP-outputs with the interface. Requests of computations are forwarded by ENGINE to MATH, which contains all the mathematical algorithms to manipulate coefficients, polynomials, ideals, modules and so on. MATH accepts LLP-requests from ENGINE, converts data from LLP-format into internal data and then performs the mathematical task in an efficient environment.

2 A Quick Tour

The CoCoA commands can be roughly divided into two classes: *simple commands* and *structured commands*. Examples of simple commands are: evaluation of an expression, assignment, and printing. Examples of structured commands are: if-then-else, for-loop, while-loop, foreach-loop, and function definition.

The expressions follow a simple syntax which is very close to the usual mathematical one.

2.1 Ring Environments

Every object in CoCoA is defined over a *base ring* which is a polynomial ring with some annotations (about ordering or weights or other things).

Each computation in CoCoA is performed in a ring-environment which consists of a base ring and a local memory where objects like polynomials or ideals can be stored.

The user can define and use several ring-environments. At any one time a particular *current ring-environment* is active within the system. The system has a default ring-environment named R and whose base ring is $Q[t, x, y, z]$.

2.2 First Examples

The simplest way to use CoCoA after starting it, is to set a ring (or use the default ring), type a command, and ask the system to execute it.

Example 2.1 To evaluate the expression $(x-3/2y+6t)^3$ in the ring $Q[t,x,y,z]$, the user just types the command:

```
(x - 3/2y + 6t)^3;
```

The output is:

```
216t^3 + 108t^2x + 18tx^2 + x^3 - 162t^2y - 54txy - 9/2x^2y +
81/2ty^2 + 27/4xy^2 - 27/8y^3
------------------------------
```

Note the multiplication operator "*" may be omitted.

Example 2.2 In this example the first two commands assign the results of the evaluation of the two expressions to the variables I and J; the third one prints out the value of J.

```
I := Ideal(x-t, y)^3;
J := Ideal(x^2-yz, xy-zt) + I;
J;
Ideal(x^2 - yz, xy - tz,  - t^3 + 3t^2x - 3tx^2 + x^3,
t^2y - 2txy + x^2y,  - ty^2 + xy^2, y^3)
------------------------------
```

Note that each command is followed by a semicolon and that each output is followed by dashes.

Example 2.3 This example shows how to define and use a ring different from the default ring.

```
A ::= Z/(5)[xy];
Use A;
(2x^2 - 1/2y)^4;
```

The first command uses the operator "::=" to define A to be the polynomial ring in the indeterminates x and y with coefficients in the field $Z/(5)$ of the integers modulo 5. The second command chooses A as the current ring. Then the expression $(2x^2-1/2y)^4$ is evaluated in A and the result is displayed.

```
x^8 - x^6y + x^4y^2 - x^2y^3 + y^4
------------------------------
```

Example 2.4 Whenever a command is just an expression, the result is automatically assigned to the CoCoAL variable `It`. This variable can be used like any other user-defined variable but one must remember that each subsequent execution of a "command-expression" will change its value.

```
Use R ::= Q[xyz];
(xy+yz)^3;
x^3y^3 + 3x^2y^3z + 3xy^3z^2 + y^3z^3
------------------------------
Der(It, y);
3x^3y^2 + 9x^2y^2z + 9xy^2z^2 + 3y^2z^3
------------------------------
```

The first command defines R to be the polynomial ring in the indeterminates x, y and z with coefficients in the field of rational numbers and makes R the current ring. The second command computes the expression $(xy + yz)^3$. The last two commands compute and display the derivative of the last computed value (`It`) with respect to y.

Example 2.5 In the next CoCoA session we use the structured command `For` to compute $(x + y)^7$ in the ring Z/nZ for $n \in \{3, 5, 7, 9\}$. Note that any text from a `--` to the end of the line is a comment.

```
For N := 3 To 9 Step 2 Do
  S ::= Z/(N)[xy]; --> define the ring S
  PrintLn Ring(S); --> display the ring S
  S :: (x+y)^7; --> evaluate the power in S and print it
  PrintLn; --> start a new line
  PrintLn
End;
```

The output is

```
Z/(3)[x,y]
x^7 + x^6y - x^4y^3 - x^3y^4 + xy^6 + y^7
```

```
Z/(5)[x,y]
x^7 + 2x^6y + x^5y^2 + x^2y^5 + 2xy^6 + y^7

Z/(7)[x,y]
x^7 + y^7

WARNING: Coeffs are not in a field
  GBasis-related computations could fail to terminate or be wrong
-------------------------------
Z/(9)[x,y]
x^7 - 2x^6y + 3x^5y^2 - x^4y^3 - x^3y^4 + 3x^2y^5 - 2xy^6 + y^7
```

2.3 Special Orderings

On this section we discuss two problems whose solutions require using a particular term-ordering. The first problem is how to find the minimal polynomial of an element of an algebraic extension of Q. The second one is how to find a cartesian representation of a space curve given parametrically.

Example 2.6 The minimal polynomial of $\frac{4\alpha-1}{\alpha^3}$ over Q, where α is a root of $x^7 - x - 1$ can be found by computing the reduced Gröbner basis of the ideal $(x^7 - x - 1, x^3y - 4x + 1)$ of the ring $Q[x, y]$ with respect to the lexicographic term-ordering with $x > y$. In CoCoA this is achieved by using the following commands:

```
Use R ::= Q[xy],Lex;
Set Indentation;
GBasis(Ideal(x^7-x-1, x^3y-4x+1));
```

The first command defines R to be the polynomial ring over the rational numbers, with indeterminates x, y and the lexicographic term-ordering (`Lex`) with $x > y$. The command `Set Indentation` forces the system to print each polynomial on a new line. Then a Gröbner basis of the ideal $(x^7 - x - 1, x^3y - 4x + 1)$ is computed and displayed. Such a basis contains a univariate polynomial in the indeterminate y which is the answer. The output is:

```
[ x - 10022553737/89893683351809y^6 + 49925279149/89893683351809y^5
+ 17282591991/89893683351809y^4 - 2197476813566/89893683351809y^3
- 3212847751937/89893683351809y^2 + 1627614492145/89893683351809y
- 52555866039552/89893683351809,
  y^7 - 5y^6 + 147y^4 + 640y^3 - 31y^2 + 2176y - 20479 ]
------------------------------
```

Example 2.7 Given the space curve $(t^{31} + t^6, t^8, t^{10})$, its cartesian equations can be found by eliminating the indeterminate t in the ideal $(t^{31} + t^6 - x, t^8 - y, t^{10} - z)$. In CoCoA this is achieved by using the following commands:

```
Use R ::= Q[xyzt];
Set Indentation;
Elim(t, Ideal(t^31+t^6-x, t^8-y, t^10-z ) );
```

The system automatically changes the ordering to an elimination term-ordering for t, performs the computation, and, finally, restores the original ordering and gives the result:

```
 Ideal( y^5 - z^4,
       z^8 + 2xy^3 - x^2yz - z^3,
       xy^4z^4 + 1/2yz^7 + 3/2x^2y^2 - x^3z - 1/2yz^2,
       y^4z^5 - y^4 + 2xy^2z - x^2z^2,
       y^2z^6 + 1/2xz^7 - 1/2x^3y - y^2z + 3/2xz^2,
       x^2y^4z^3 + 3y^3z^5 + 2xyz^6 - x^4 - 3y^3 + 4xyz )
------------------------------
```

If one wishes to see the entire Gröbner basis with respect to the elimination term-ordering for t used by the system in the previous computation, then it suffices to execute the commands:

```
Use R ::= Q[xyzt], Elim(t);
Set Indentation;
GBasis( Ideal(t^31+t^6-x, t^8-y, t^10-z ) );
```

The first command defines R to be the polynomial ring in the indeterminates x, y, z, and t with coefficients in the ring of rational

numbers and makes R the current ring. Moreover the "ring modifier" `Elim(t)` in the ring definition endows the ring R with an elimination term-ordering for the indeterminate t.

After the execution of these commands, one gets the following result:

```
[ yt^2 - z,
  y^5 - z^4,
  t^6 + z^3t - x,
  z^4t + y^2 - xz,
  y^2t - xzt + y^4z,
  xt^2 - x^2z^2t + xy^4z^2 + yz^5 - y,
  z^8 + 2xy^3 - x^2yz - z^3,
  xy^4z^4 + 1/2yz^7 + 3/2x^2y^2 - x^3z - 1/2yz^2,
  zt^2 + 2xz^3t - y^4z^3 - x^2,
  y^4z^5 - y^4 + 2xy^2z - x^2z^2,
  xyz^3t - 1/2z^7 - 1/2x^2y + 1/2z^2,
  x^2z^3t - 2/3xy^4z^3 - 1/3yz^6 - 1/3x^3 + 1/3yz,
  x^2yt - z^2t - y^2z^3 - xz^4,
  x^3t - yzt - x^2y^4 - y^3z^2 - xyz^3,
  y^2z^6 + 1/2xz^7 - 1/2x^3y - y^2z + 3/2xz^2,
  x^2y^4z^3 + 3y^3z^5 + 2xyz^6 - x^4 - 3y^3 + 4xyz ]
-------------------------------
```

You may see the term-ordering of the current ring by using the function `Ord`. The command `Unset Indentation` prevents the system from printing each entry of the matrix on a different line.

```
Unset Indentation;
Ord();
```

The result is

```
Mat[
  [0, 0, 0, 1],
  [1, 1, 1, 0],
  [0, 0, -1, 0],
  [0, -1, 0, 0]
]
-------------------------------
```

2.4 Playing with Several Rings

Two or more rings may be defined. One of these rings is the current ring, but the user may also compute in a ring different from the current ring. To do this, the user must specify in which ring an expression should be evaluated.

Example 2.8 In this example we define two different rings R and S, and we choose R as the current ring. The coefficient ring of S is $Z/5Z$. The ordering on R is `DegRevLex` by default. Then we define an ideal J and a polynomial F in R. What we want to do is to print an expression defined over S if some conditions hold in the ring R. More precisely: if the normal form of F with respect to the ideal obtained by eliminating t from the ideal J is zero, then we compute the expressions $(a+t)^{I+1}$ in the ring S, for $I = 3, 5, 7, 9$.

```
S ::= Z/(5)[tabc],Lex;
Use R ::= Q[txyz];
J := Ideal(t^31-t^6-x, t^8-y, t^10-z);
F := y^5-z^4;
If NF(F, Elim(t,J)) = 0 Then
  For I := 3 To 9 Step 2 Do
    PrintLn;
    S :: (a+t)^(I+1);
  End
End;
```

The output is:

```
t^4 - t^3a + t^2a^2 - ta^3 + a^4
t^6 + t^5a + ta^5 + a^6
t^8 - 2t^7a - 2t^6a^2 + t^5a^3 + t^3a^5 - 2t^2a^6 - 2ta^7 + a^8
t^10 + 2t^5a^5 + a^10
------------------------------
```

2.5 Functions

User defined functions may have any number of parameters of any type and may return a result. They contain sequences of CoCoA commands.

Calling a user defined function is achieved by typing its name followed by a list of arguments, possibly empty, in parentheses. Recursive function are also allowed.

The following code provides a naive implementation of the primality test for an integer.

```
Define Prime(X)
  If Type(X) <> INT Then Return Error('Expected INT') End;
  I := 2;
  While I*I <= X  Do
    If Div(X,I)*I = X Then Return FALSE End;
    I := I + 1
  End;
  Return TRUE
End;
```

In the following for-loop we use the function *Prime* defined above to check the primality of the integer I; in that case we define a polynomial ring P over the ring of the integers modulo I and compute an expression in P.

```
For I := 3 To 10 Do
  If Prime(I) Then
    P ::= Z/(I)[xyzw];
    PrintLn 'I = ', I;
    P :: (x+y+z+w)^I;
    PrintLn
  End
End;
```

The output is:

```
I = 3
x^3 + y^3 + z^3 + w^3
I = 5
x^5 + y^5 + z^5 + w^5
I = 7
x^7 + y^7 + z^7 + w^7

------------------------------
```

2.6 Indeterminates with Indices

The next two examples show how to use indeterminates with indices.

Example 2.9 This example produces the leading term ideal of the ideal generated by three "generic" polynomials of degree 2 in the indeterminates $x[1]$, $x[2]$, $x[3]$, and $x[4]$ with respect to the lexicographic term-ordering.

```
Use R ::= Z/(32003)[x[1..4]],Lex;
F := DensePoly(2);
-- DensePoly(n) returns the polynomial which is the sum of
-- all the terms of degree n of the current ring
L := [ Randomized(F) | I In 1..3 ];
-- Randomized(F) randomizes the coefficients of F
LT(Ideal(L));
```

The output is:

```
Ideal( x[1]^2, x[1]x[2], x[1]x[3], x[2]^3, x[1]x[4]^2, x[2]^2x[3],
x[2]^2x[4]^2, x[2]x[3]^3, x[2]x[3]^2x[4]^2, x[2]x[3]x[4]^4,
x[2]x[4]^6, x[3]^8 )
-------------------------------
```

Example 2.10 In this example we compute the equations of the sub-algebra generated by the 2×2 minors of generic $2 \times N$ matrices ($N = 3, 4, 5$).

The following function defines the sub-algebra.

```
Define Det_SubAlgebra(N)
  L := [];
  For C1 := 1 To N-1 Do
    For C2 := C1+1 To N Do
      P := y[C1,C2]-(x[1,C1] x[2,C2] - x[2,C1] x[1,C2]);
      Append(L,P)
    End
  End;
  Return Ideal(L)
End;
```

The following function compute and prints out the equations of the sub-algebra.

```
Define Det_SubAlgebra_Print(N)
  J := Det_SubAlgebra(N);
  PrintLn NewLine,'N = ',N;
  PrintLn 'Sub-algebra equations:';
  PrintLn Gens(Elim(x,J))
End;
```

The following for-loop calls Det_SubAlgebra_Print for N=3,4,5.

```
Set Indentation;
For N := 3 To 5 Do
  S ::= Z/(32003)[y[1..(N-1),2..N]x[1..2,1..N]];
  S :: Det_SubAlgebra_Print(N);
End;
```

The output is:

```
N = 3
Sub-algebra equations:
[ 0 ]

N = 4
Sub-algebra equations:
[ y[1,4]y[2,3] - y[1,3]y[2,4] + y[1,2]y[3,4] ]

N = 5
Sub-algebra equations:
[ y[2,5]y[3,4] - y[2,4]y[3,5] + y[2,3]y[4,5],
  y[1,5]y[3,4] - y[1,4]y[3,5] + y[1,3]y[4,5],
  y[1,5]y[2,4] - y[1,4]y[2,5] + y[1,2]y[4,5],
  y[1,5]y[2,3] - y[1,3]y[2,5] + y[1,2]y[3,5],
  y[1,4]y[2,3] - y[1,3]y[2,4] + y[1,2]y[3,4] ]
```

2.7 Rapid Prototyping

The CoCoA language can be used for "rapid prototyping" of algorithms.

Example 2.11 The following is not an efficient algorithm for the computation of the ideal of points, but it works.

The example shows an advanced use of error handling. The function IdealOfPoint may return two different errors.

```
Define IdealOfPoint(P)
  -- first check if the number of the coordinates is correct
  If Len(P) <> NumIndets Then
    Return Error('Wrong number of indeterminates')
  End;
  -- then compute the index of the first non zero component of P
  F := 1;
  While F <= NumIndets And P[F]=0 Do
    F := F + 1
  End;
  -- if all components of P are 0 then reject P
  If F>Len(P) Then Return Error('Not a projective point') End;
  -- otherwise return the ideal of P
  L := [ Indet(K)*P[F] - Indet(F)*P[K] | K In 1..NumIndets And K <> F ];
  -- Indet(N) gives the N-th indeterminate of the current ring
  Return Ideal(L)
End;
```

The function IdealOfPoints(L) returns the ideal of the points in the list L through multiple calls to IdealOfPoint. If IdealOfPoint fails, then the command "catch" intercepts the error, and IdealOfPoints immediately returns the error together with the bad point.

```
Define IdealOfPoints(L)
  -- Ideal of N points in the array L
  Catch I := IdealOfPoint(Head(L)) In E End;
  If Type(E) = ERROR Then Print Head(L), ': '; Return E End;
  Foreach P In Tail(L) Do
    Catch J := IdealOfPoint(P) In E End;
    If Type(E) = ERROR Then Print P, ': '; Return E End;
    I := Intersection(I,J)
  End;
  Return I
End;
```

The following commands prepare a list of points $[P_1, ..., P_5]$ in the projective space $\mathbf{P}^3$.

```
Use R ::= Q[xyzw];
A := [
  [1,0,1,0];
  [0,1,1,0];
  [1,1,1,1];
  [0,0,0,0];
  [0,1,0,3]
];
```

The for-loop computes the ideal of the points $[P_1, ..., P_i]$, for $i = 1, \ldots, 5$.

```
For I := 1 To Len(A) Do
  IdealOfPoints(First(A,I));
  -- First(A,I) gives the list of the first I elements of the list A
  PrintLn;
End;
```

The output is

```
Ideal(y,  - x + z, w)
Ideal(w, x + y - z, y^2 - yz)
Ideal(x + y - z - w, zw - w^2, y^2 - yz, yw - w^2)
[0, 0, 0, 0]: ERROR [stdin] Not a projective point
-------------------------------
```

Since P_4 is not a projective point, the loop terminates with an error generated by the function IdealOfPoint.

2.8 Minimal Free Resolutions

Example 2.12 In this example we present some new features available in the forthcoming version of CoCoA (see Sect. 3). We compute the minimal free resolution of the ideal I generated by the 2 by 2 minors of a catalecticant matrix A, using the interactive environment.

First we load the 'gb' package. Then we define the ideal I and start the computation of its minimal free resolution using the Hilbert-driven algorithm described in [CDNR].

```
<< 'gb.pkg';

Use R ::= Z/(32003)[z[0..3,0..3,0..3]];

A := Mat[
  [z[3,0,0], z[2,1,0], z[2,0,1]],
  [z[2,1,0], z[1,2,0], z[1,1,1]],
  [z[2,0,1], z[1,1,1], z[1,0,2]],
  [z[1,2,0], z[0,3,0], z[0,2,1]],
  [z[1,1,1], z[0,2,1], z[0,1,2]],
  [z[1,0,2], z[0,1,2], z[0,0,3]]
];

I := Ideal(Minors(2,A));
GB.Init_Computation(I,'Res_Hilbert');
```

Now we ask the system to perform 1000 steps of computation and then we explore the partial result.

```
GB.Steps(I,1000);
GB.Get_Res(I);
0 --> R^159(-5) --> R^189(-4) --> R^105(-3) --> R^27(-2)
-------------------------------

GB.Res_Report(I);
-----------------------------------------------------------------
Minimal Pairs,              :   754
  Groebner Pairs            :    48
  Minimal (Type S)          :   706
    Minimal (Type Smin)     :   453
    Minimal (Type S0)       :   253
      H-Killed (Type S0)    :     9
      Hard (Type S0)        :   244
-----------------------------------------------------------------
```

Finally we complete the computation and see the result.

```
GB.Complete(I);
GB.Get_Res(I);
0 --> R(-9) --> R^27(-7) --> R^105(-6) --> R^189(-5) --> R^189(-4)
--> R^105(-3) --> R^27(-2)
-------------------------------
```

3 Work in Progress and the Future

The next version (3.1) of CoCoA will present many new features.

We have designed an interactive environment which can be customized in such a way that the user can, for instance, execute Gröbner-related computations step by step and explore every partial result. In particular it can be used with the new algorithms for the computation of the minimal free resolution of a module [CDNR].

The Input/Output management has been completely redesigned. It is now based on the notion of "device" which is characterized by its type (file, string, ...) and the protocol for the communication with the interface.

The CoCoA library has been organized into "packages", a new feature of the language which allows better modularity and solves the problem of "name clashes".

Some work is in progress on the interfaces. The next version will feature an enhanced Macintosh interface and a WWW interface. The latter has been designed to allow the user to interact with a CoCoA "server" using any web browser; see [CD] for more details.

In the near future many features are going to be added to the system; in particular an advanced package of Linear Algebra, an implementation of the Tangent Cone Algorithm, Gröbner bases over the integers, homomorphisms, more general rings (for instance quotients and localizations of polynomial rings), an interface to OpenMath [ADS] (by means of a new protocol for the devices), and a graphical interface for Windows 95.

References

[ADS] J. Abbott, A. Diaz, R. Sutor, *A Report on OpenMath*, SIGSAM Bulletin, 30(1996), 21–24.

[AL] W.W. Adams, P. Loustaunau, *An Introduction to Gröbner Bases*, Graduate Studies in Mathematics, AMS, 1994.

[B] A.M. Bigatti, *Computations of Hilbert-Poincaré Series*, J. Pure Appl. Algebra, to appear.

[BS] D. Bayer, M. Stillman, *Macaulay*: A system for computation in algebraic geometry and commutative algebra, 1992. Available via anonymous ftp from `math.harvard.edu`.

[CD] A. Capani, G. De Dominicis, *Web Algebra*, In Proc. of WebNet 96. Association for the Advancement of Computing in Education (AACE) Charlottesville, USA, 1996.

[CDNR] A. Capani, G. De Dominicis, G. Niesi, L. Robbiano, *Computing Minimal Finite Free Resolutions*, J. Pure Appl. Algebra, to appear.

[CDR] M. Caboara, G. De Dominicis, L. Robbiano, *Multigraded Hilbert Functions and Buchberger Algorithm*, In Y.N. Lakshman, editor, *Proc. ISSAC '96*, 72–78, New York, 1996. ACM Press.

[CNR] A. Capani, G. Niesi, L. Robbiano, CoCoA, *a system for doing Computations in Commutative Algebra*, (1995). Available via anonymous ftp from `lancelot.dima.unige.it`.

[E] D. Eisenbud *Commutative Algebra with a View Toward Algebraic Geometry*, Springer Graduate Texts in Mathematics **150**, (1995).

[GMNRT] A. Giovini, T. Mora, G. Niesi, L. Robbiano, C. Traverso, *"One sugar cube, please" or selection strategies in the Buchberger algorithm*, In Editor Stephen M. Watt, editor, *Proc. ISSAC '91*, 49–54, New York, 1991. ACM Press.

[GN] A. Giovini and G. Niesi, *CoCoA: A user-friendly system for commutative algebra*, In *Design and Implementation of Symbolic Computation Systems – International Symposium DISCO'90*, Lecture Notes in Comput. Sci., 429, 20–29, Berlin, 1990. Springer Verlag.

[GPS] G. M. Greuel, G. Pfister, H. Schönemann, *Singular:*, A System for Computation in Algebraic Geometry and Singular Theory. De-

partment of Mathematics, 1995. Available via anonymous ftp from `helios.mathematik.uni-kl.de`.

[GS] D. Grayson, M. Stillman, *Macaulay 2*, 1996. Available via anonymous ftp from `math.uiuc.edu`.

[T] C. Traverso *Hilbert functions and the Buchberger algorithm*, J. Symbolic Computation. To appear.

Antonio Capani, Gianfranco Niesi,
Lorenzo Robbiano,
Department of Mathematics,
University of Genova
Via Dodecaneso, 35
I-16146 Genova, Italy

Received 28 November, 1996