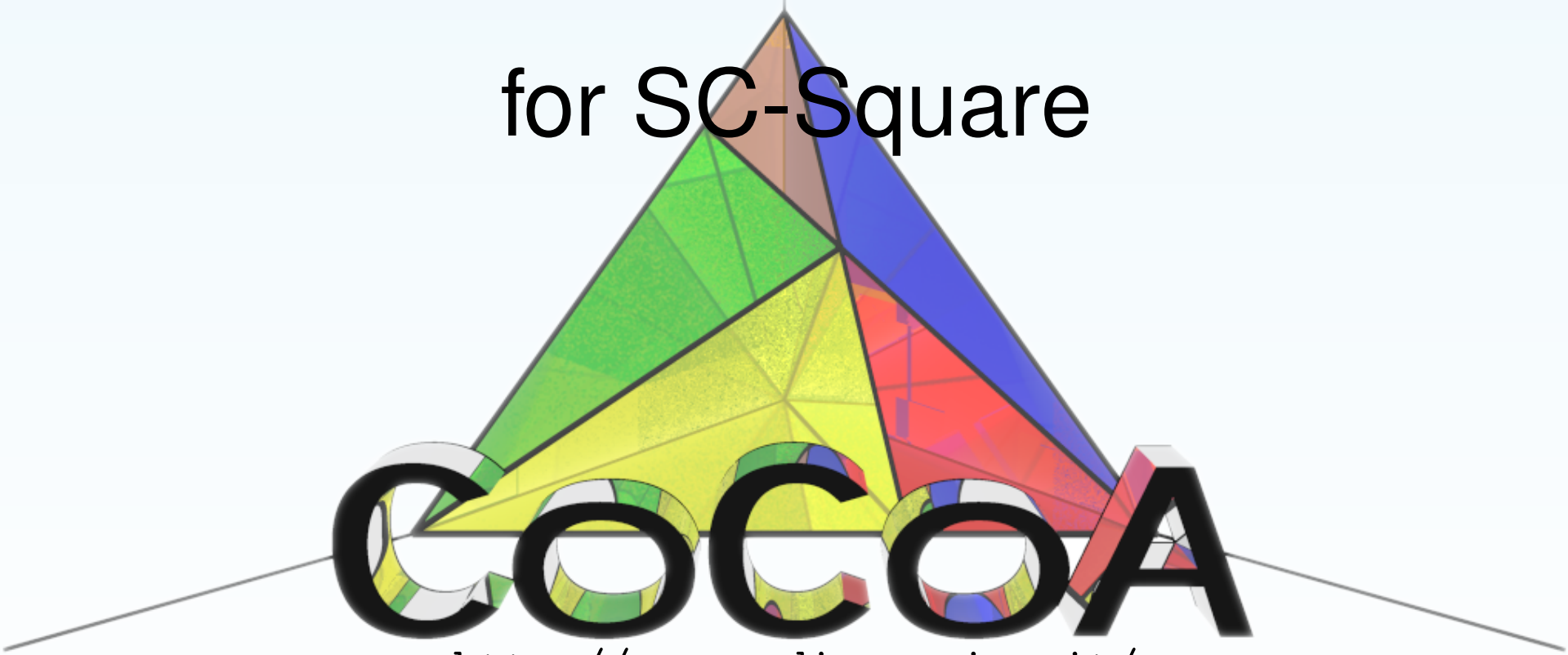
New in CoCoA-5.2.2 and CoCoALib-0.99560

for SC-Square



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

J. Abbott & **A.M. Bigatti**

INdAM-COFUND Marie Curie Fellow & Università di Genova

The CoCoA project began in 1987 under the lead of Prof. L. Robbiano. The aim was to create a *mathematician-friendly* software laboratory for studying Computational Commutative Algebra (multivariate polynomials).

Available as open-source software: 2 main components

- CoCoALib — C++ library
- CoCoA-5 — interactive system

Main new features in CoCoALib-0.99560 and CoCoA-5.2.2

- interrupt handling and timeout capability
- global verbose option (user-settable verbosity level)
- minimal polynomials in quotients by 0-dim ideals
  - $\longrightarrow$ radical, primary decomposition, tests for maximal or primary
- efficient hypersurface implicitization
- improved Gröbner fan operations (needs `GFanLib`)
- prototype interface with `MathSAT`

Interrupt Mechanism: example C++ program

```
void LongComputation()
{ ...
  for (int i=1; i < n; ++i) // MAIN LOOP
  { CheckForInterrupt("LongComputation"); // arg gives context info
    ...
  }
  ...
}

void program()
{ ...
  SignalWatcher MonitorSIGINT(SIGINT); // RAII object
  try { LongComputation(); }
  catch (const InterruptedBySignal& intr)
  { PrintInFrame(cout, intr); /*handle the interrupt*/ }
}
```

Timeout Mechanism: example C++ program

```
// New fn to add "timeout" feature to existing fn:
vector<RingElem> RGBasisTimeout(const ideal& I, double Tmax)
{
  CpuTimeLimit timeout(Tmax);
  return ReducedGBasis(I); // <-- this fn must call CheckForInterrupt
}

void program()
{
  GlobalManager CoCoAFoundations;
  ...
  ideal I = ideal(f1,f2,f3);
  // Next line will throw InterruptedByTimeout if timeout occurs:
  vector<RingElem> RGB = RGBasisTimeout(I, 3.0); // max 3.0 seconds
}
```

Timings for `MinPolyQuot`

- 0-dim ideal (gen. by dense random polys)
- dense random element in span of "quotient basis"
- in ring $\mathbb{Q}[x, y, z]$; random coeffs are in range $-9 \ldots 9$

Deg	Time	Eliminate
3	1.4	45
4	28	14000+
5	295	—
6	2050	—
7	11600	—

Time is worst of 5 trials

Timings for `PrimaryDecomposition`

Randomly generated zero-dimensional ideals in $\mathbb{Q}[x, y, z]$

Favourable case

$\langle 29x^3y^3+94yz^5+99x^4, -36y^3z^3-67z^6-3x^4y, 19x^3y^3+97x^2z^2-85y^2\rangle$
CoCoA took 5.6s
Singular took 1600s

Unfavourable case

$\langle -11x^5 - 10x^4y + 12y^3z, -14z^4 + 15y^3 + 13yz, 9xz^4 + x^3y + 2y^3z\rangle$

CoCoA took 45s
Singular took 3.3s

Web site: http://cocoa.dima.unige.it/

Interim downloads:

- CoCoALib v. 0.99555 (August 2017)
- CoCoA v. 5.2.1 (August 2017)

Imminent stable downloads:

- CoCoALib v. 0.99560 (expected September 2017)
- CoCoA v. 5.2.2 (expected September 2017)

Also... redmine (bug/feature tracker), & other useful bits and bobs.

CoCoA and MathSAT

Two actors of the **SC^2 project:**

- **MathSAT**: SMT solver (Trento)
- **CoCoALib**/**CoCoA-5**: Computer Algebra System (Genova)

One SC^2 deliverable: prototype of communication between them

1. **MathSAT** using **CoCoALib**'s `RingTwinFloat` arithmetic (instead of arbitrary precision rationals, GMP `mpq`)
2. **MathSAT** using **CoCoALib** for systems of polynomial inequalities (future! hopefully)
3. enabling **CoCoALib** to call some of **MathSAT**'s functions
4. using **CoCoA-5** as a handy interface for **MathSAT** (through CoCoALib)

(3) CoCoAlib calling MathSAT: C++ program excerpt

```
ring P = NewPolyRing(QQ, symbols("x,y"));
MathSAT::env E;

matrix M = NewDenseMat(QQ, 2, 3); // (2x3 matrix)
SetEntry(M,0,0,  3);   SetEntry(M,0,1,  1);   SetEntry(M,0,2, -2);
SetEntry(M,1,0,  1);   SetEntry(M,1,1, -1);   SetEntry(M,1,2,  0);
MathSAT::AddLeq0(E, M);       // 3*x -y -2 <= 0
                              //   x -y    <= 0
M = NewDenseMat(QQ, 1, 3);  SetEntry(M,0,0, 1);  // (1x3 matrix)
MathSAT::AddNeq0(E, M);       //   x       != 0

// polynomial syntax:
RingElem x(P,"x"), y(P,"y");
MathSAT::AddLeq0(E, x -9);    //   x    -9 <= 0
MathSAT::AddEq0(E,  x -y);    //   x -y     = 0
MathSAT::AddLt0(E,  2*x-1);   // 2*x    -1  < 0

cout << "A solution is " << MathSAT::LinSolve(E) << endl; //-> 1/4, 1/4
```

(4) CoCoA-5 calling MathSAT: interactive call

We represent a system as a record of matrices (first prototype):

```
system :=
  record[leq0 := matrix([[1,2,3, 4],  //   x +2*y +3*z +4 <= 0
                         [9,8,7, 0]]),// 9*x +8*y +7*z    <= 0
         neq0 := matrix([[1,0,0, 0]]) //   x              <> 0
        ];

sol := MSatLinSolve(system);  sol;
// gives  [[1], [4/5], [-11/5]]

// verify:
sol1 := ConcatVer(sol, matrix([[1]])); //-->  [[1], [4/5], [-11/5], [1]]
system.leq0 * sol1;  //-->  0, 0   <= 0
system.neq0 * sol1;  //-->  1     <> 0
```

Continuation:

```
// solution was  [[1], [4/5], [-11/5]]
// now we add new contraints:
system.eq0 := RowMat([1,1,0, 4]);   //  x +y +4 = 0
system.lt0 := RowMat([0,1,0, 0]);   //     y    < 0

sol := MSatLinSolve(system);  sol;
// gives  [[-2], [-2], [-2/7]])

// verify:
sol1 := ConcatVer(sol, RowMat([1]));  //-->  [[-2], [-2], [-2/7], [1]]
system.leq0 * sol1;  //-->  -20/7, -36  <= 0
system.neq0 * sol1;  //-->  -2          <> 0
system.eq0  * sol1;  //-->   0           = 0
system.lt0  * sol1;  //-->  -2           < 0
```