Current Challenges in the Development of Open Source Computer Algebra Software

Wolfram Decker

TU Kaiserslautern

IMCS 2016, Berlin, July 13, 2016

Algorithmic and Experimental Methods
in Algebra, Geometry, and Number Theory
DFG Priority Project SPP 1489

Some Basic Statements

- Methods from computer algebra are now firmly established in the toolbox of the pure mathematician.

Some Basic Statements

- Methods from computer algebra are now firmly established in the toolbox of the pure mathematician.
- Particular fruitful interactions unfold between computer algebra and group and representation theory, algebraic geometry, polyhedral geometry, and number theory.

Some Basic Statements

- Methods from computer algebra are now firmly established in the toolbox of the pure mathematician.
- Particular fruitful interactions unfold between computer algebra and group and representation theory, algebraic geometry, polyhedral geometry, and number theory.

GAP: group and representation theory, general purpose high level interpreted programming language.

We Build on Four Cornerstones

GAP: group and representation theory, general purpose high level interpreted programming language.

SINGULAR: polynomial computations in algebraic geometry, commutative algebra, and singularity theory.

We Build on Four Cornerstones

GAP: group and representation theory, general purpose high level interpreted programming language.

SINGULAR: polynomial computations in algebraic geometry, commutative algebra, and singularity theory.

polymake: convex polytopes, polyhedral fans, simplicial complexes, related objects from combinatorics & geometry.

We Build on Four Cornerstones

GAP: group and representation theory, general purpose high level interpreted programming language.

SINGULAR: polynomial computations in algebraic geometry, commutative algebra, and singularity theory.

polymake: convex polytopes, polyhedral fans, simplicial complexes, related objects from combinatorics & geometry.

ANTIC: number theoretic software, computations in and with number fields and generic finitely presented rings.

We Build on Four Cornerstones

GAP: group and representation theory, general purpose high level interpreted programming language.

SINGULAR: polynomial computations in algebraic geometry, commutative algebra, and singularity theory.

polymake: convex polytopes, polyhedral fans, simplicial complexes, related objects from combinatorics & geometry.

ANTIC: num theo-
retic softw a-
tions in
ber field
finitely p

Created within SPP 1489

GAP

Powerful language

- General purpose yet ideally suited to algebraic computation;
- rapid implementation of mathematical algorithms;
- intelligent GAP objects and algorithms.

Example

```
gap> grp := PSL(5,3);;
gap> a:=PseudoRandom(grp);; b:=PseudoRandom(grp);;
gap> Size( Group(a,b) );
237783237120 10 msec
```

Parametrizing Rational Curves in SINGULAR

Example

```
> ring R = 0, (x,y,z), dp;
> poly f = x5+10x4y+20x3y2+130x2y3-20xy4+20y5-2x4z-40x3yz-150x2y2z
           -90xy3z-40y4z+x3z2+30x2yz2+110xy2z2+20y3z2;
> LIB "paraplanecurves.lib";
> genus(f); // may require blow-ups
  0
> paraPlaneCurve(f); // requires normalization, integral bases
```

Parametrizing Rational Curves in SINGULAR

Example

```
> ring R = 0, (x,y,z), dp;
> poly f = x5+10x4y+20x3y2+130x2y3-20xy4+20y5-2x4z-40x3yz-150x2y2z
           -90xy3z-40y4z+x3z2+30x2yz2+110xy2z2+20y3z2;
> LIB "paraplanecurves.lib";
> genus(f); // may require blow-ups
  0
> paraPlaneCurve(f); // requires normalization, integral bases
```

Remark

Basic workhorse: Buchberger's algorithm for computing Gröbner bases and syzygies.

Parametrizing Rational Curves in SINGULAR

Example

```
> ring R = 0, (x,y,z), dp;
> poly f = x5+10x4y+20x3y2+130x2y3-20xy4+20y5-2x4z-40x3yz-150x2y2z
           -90xy3z-40y4z+x3z2+30x2yz2+110xy2z2+20y3z2;
> LIB "paraplanecurves.lib";
> genus(f); // may require blow-ups
  0
> paraPlaneCurve(f); // requires normalization, integral bases
```

Remark

Basic workhorse: Buchberger's algorithm for computing Gröbner bases and syzygies.

May require to work over quadratic field extension.

Example

Parametrizing Rational Curves in SINGULAR

Example

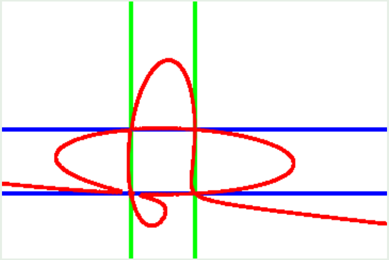
We consider the degree-5 curve with equation

$$x^5 + 10x^4y + 20x^3y^2 + 130x^2y^3 - 20xy^4 + 20y^5 - 2x^4z$$
$$- 40x^3yz - 150x^2y^2z - 90xy^3z - 40y^4z + x^3z^2 + 30x^2yz^2$$
$$+ 110xy^2z^2 + 20y^3z^2 = 0.$$

Example

We consider the degree-5 curve with equation

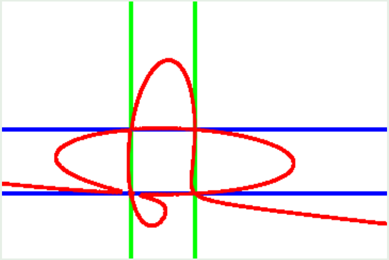
$$x^5 + 10x^4y + 20x^3y^2 + 130x^2y^3 - 20xy^4 + 20y^5 - 2x^4z$$
$$- 40x^3yz - 150x^2y^2z - 90xy^3z - 40y^4z + x^3z^2 + 30x^2yz^2$$
$$+ 110xy^2z^2 + 20y^3z^2 = 0.$$



Example

We consider the degree-5 curve with equation

$$x^5 + 10x^4y + 20x^3y^2 + 130x^2y^3 - 20xy^4 + 20y^5 - 2x^4z - 40x^3yz - 150x^2y^2z - 90xy^3z - 40y^4z + x^3z^2 + 30x^2yz^2 + 110xy^2z^2 + 20y^3z^2 = 0.$$



Genus Formula. $p_g(C) = p_a(C) - \delta(C) = p_a(C) - \sum_{P \in \mathrm{Sing}(\Gamma)} \delta_P(C)$

Some Basic Statements

- A decisive feature of current developments is that more and more of the abstract mathematical concepts are made constructive, with interdisciplinary methods playing a significant role.

The Four Cornerstones

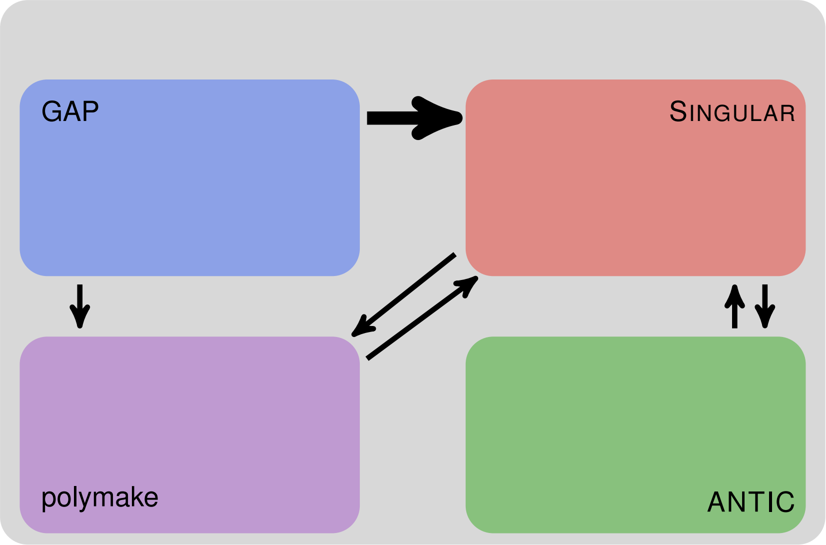
GAP: group and representation theory, general purpose high level interpreted programming language.

SINGULAR: polynomial computations in algebraic geometry, commutative algebra, and singularity theory.

polymake: convex polytopes, polyhedral fans, simplicial complexes, related objects from combinatorics & geometry.

ANTIC: number theoretic software, computations in and with number fields and generic finitely presented rings.

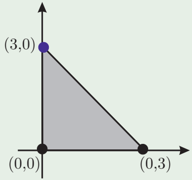
Interfaces created within SPP 1489

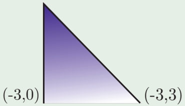


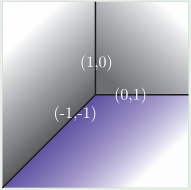
Example: Using POLYMAKE IN SINGULAR

Example

```
> LIB "polymake.so";
   Welcome to polymake
   Copyright (c) 1997-2012
   Ewgenij Gawrilow, Michael Joswig (TU Darmstadt)
   http://www.polymake.org
   // ** loaded polymake.so
> ring r = 0,(x,y),dp; poly g = x3+y3+1;
> polytope p = newtonPolytope(g);
> fan F = normalFan(p); F;
RAYS:
-1 -1 #0
 0  1 #1
 1  0 #2
MAXIMAL_CONES:
{0 1} #Dimension 2
{0 2}
{1 2}
```







- The design and further development of successful open source computer algebra software is always driven by intended applications, irrespective of whether these applications lie within or outside of mathematics.

A First Motivating Example: Surfaces of General Type With $p_g = 0$

"Can a computer classify all surfaces of general type with $p_g = 0$?"

- David Mumford, Montreal, 1980 -

A First Motivating Example: Surfaces of General Type With $p_g = 0$

"Can a computer classify all surfaces of general type with $p_g = 0$?"

- David Mumford, Montreal, 1980 -

The problem

A minimal surface of general type with $p_g = 0$ satisfies $1 \leq K^2 \leq 9$ (Bogomolov-Miyaoka-Yau inequality).

A First Motivating Example: Surfaces of General Type With $p_g = 0$

"Can a computer classify all surfaces of general type with $p_g = 0$?"

- David Mumford, Montreal, 1980 -

The problem

A minimal surface of general type with $p_g = 0$ satisfies $1 \leq K^2 \leq 9$ (Bogomolov-Miyaoka-Yau inequality). For each $1 \leq n \leq 9$, there is the Gieseker moduli space representing the isomorphism classes of surfaces with $K^2 = n$.

A First Motivating Example: Surfaces of General Type With $p_g = 0$

"Can a computer classify all surfaces of general type with $p_g = 0$?"

- David Mumford, Montreal, 1980 -

The problem

A minimal surface of general type with $p_g = 0$ satisfies $1 \leq K^2 \leq 9$ (Bogomolov-Miyaoka-Yau inequality). For each $1 \leq n \leq 9$, there is the Gieseker moduli space representing the isomorphism classes of surfaces with $K^2 = n$. At current state, the complete description of these moduli spaces is wide open.

A First Motivating Example: Surfaces of General Type With $p_g = 0$

"Can a computer classify all surfaces of general type with $p_g = 0$?"

- David Mumford, Montreal, 1980 -

The problem

A minimal surface of general type with $p_g = 0$ satisfies $1 \leq K^2 \leq 9$ (Bogomolov-Miyaoka-Yau inequality). For each $1 \leq n \leq 9$, there is the Gieseker moduli space representing the isomorphism classes of surfaces with $K^2 = n$. At current state, the complete description of these moduli spaces is wide open.

Work by Castelnuovo, Enriques, Godeaux, Campedelli, Miyaoka, Reid and students, Beauville, Bauer-Catanese and students, and many more.

A First Motivating Example: Surfaces of General Type With $p_g = 0$

Example (The case $K^2 = 1$: Numerical Godeaux surfaces)

For such a surface X, it is known that $\pi_1(X)^{ab}$ is cyclic of order at most 5, and constructions have been given for each choice of order.

A First Motivating Example: Surfaces of General Type With $p_g = 0$

Example (The case $K^2 = 1$: Numerical Godeaux surfaces)

For such a surface X, it is known that $\pi_1(X)^{ab}$ is cyclic of order at most 5, and constructions have been given for each choice of order. It is conjectured that there is precisely one irreducible family of surfaces for each order, and that in each case $\pi_1(X) \cong \pi_1(X)^{ab}$.

A First Motivating Example: Surfaces of General Type With $p_g = 0$

Example (The case $K^2 = 1$: Numerical Godeaux surfaces)

For such a surface X, it is known that $\pi_1(X)^{ab}$ is cyclic of order at most 5, and constructions have been given for each choice of order. It is conjectured that there is precisely one irreducible family of surfaces for each order, and that in each case $\pi_1(X) \cong \pi_1(X)^{ab}$.

Experimental approach to solve the conjecture aims at constructing random points in the Gieseker moduli space and study the related geometry using the computer.

A First Motivating Example: Surfaces of General Type With $p_g = 0$

Example (The case $K^2 = 1$: Numerical Godeaux surfaces)

For such a surface X, it is known that $\pi_1(X)^{ab}$ is cyclic of order at most 5, and constructions have been given for each choice of order. It is conjectured that there is precisely one irreducible family of surfaces for each order, and that in each case $\pi_1(X) \cong \pi_1(X)^{ab}$.

Experimental approach to solve the conjecture aims at constructing random points in the Gieseker moduli space and study the related geometry using the computer. This requires methods from computational algebraic geometry, topology, and group theory.

A First Motivating Example: Surfaces of General Type With $p_g = 0$

Example (The case $K^2 = 1$: Numerical Godeaux surfaces)

For such a surface X, it is known that $\pi_1(X)^{ab}$ is cyclic of order at most 5, and constructions have been given for each choice of order. It is conjectured that there is precisely one irreducible family of surfaces for each order, and that in each case $\pi_1(X) \cong \pi_1(X)^{ab}$.

Experimental approach to solve the conjecture aims at constructing random points in the Gieseker moduli space and study the related geometry using the computer. This requires methods from computational algebraic geometry, topology, and group theory.

Joint project with Frank-Olaf Schreyer and Isabel Stenger.

A First Motivating Example: Surfaces of General Type With $p_g = 0$

Why is this computationally hard from the very beginning?

```
> Rextension;
//   characteristic : 0
//   1 parameter    : a
//   minpoly        :
(24873879473832817299558394474990433025260537858429700*a^8
+412197480758832021377448558823165698794277118212212070*a^7
+625366891611244986389942014312773193649951168354090190*a^6
-436561073546512334083477547357856090524552855592558795*a^5
-914947642504230095779800456657440020138074539186145912*a^4
-2227325279423247966617649640155997715235288113299887954*a^3
+2312070077580715288467637707530192772778088469836344950*a^2
+1366053134215201364075122803745127996518986576734818612*a
-1156759915557562158859054495379551857229358735237021536)
//   number of vars : 12
```

- Through computer algebra systems, a large treasure of mathematical knowledge becomes accessible to and can also be applied by non-experts.

Citation Record (data from `https://zbmath.org`)

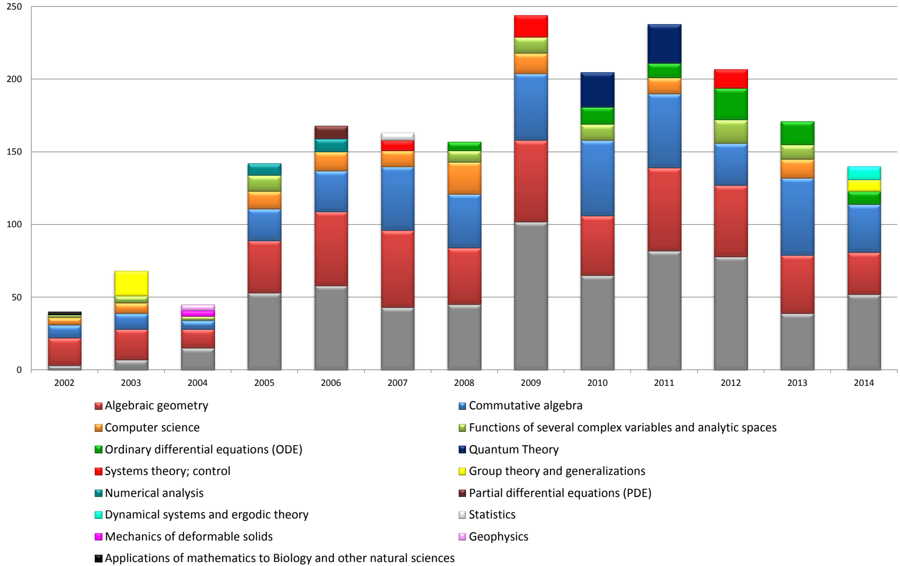
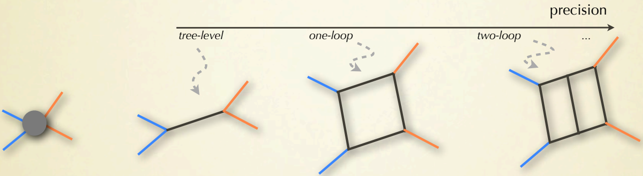
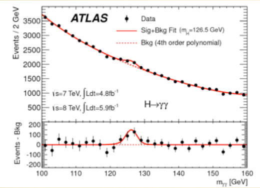
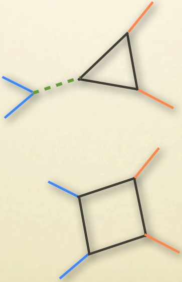


Figure: Top five MSC areas for SINGULAR per year, remaining areas in grey

Scattering ~ Feynman Diagrams







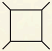
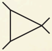
- same signature in the experiment
- only one of them contributes to the little bump
- precision calculation to distinguish them:

  signal vs background analysis

A Second Motivating Example: High Energy Physics

One-Loop Scattering Amplitudes

- n-particle Scattering: $1+2 \to 3+4+\ldots+n$
- Reduction to a Scalar-Integral Basis Passarino-Veltman

$$= \sum_{10^2-10^3} \int d^D\ell \, \frac{\ell^\mu \ell^\nu \ell^\rho \ldots}{D_1 D_2 \ldots D_n} = c_4$$  $$+ c_3$$  $$+ c_2$$  $$+ c_1$$ 

- Known: Master Integrals

 $= \int d^D\ell \, \frac{1}{D_1 D_2 D_3 D_4}$,  $= \int d^D\ell \, \frac{1}{D_1 D_2 D_3}$,  $= \int d^D\ell \, \frac{1}{D_1 D_2}$,  $= \int d^D\ell \, \frac{1}{D_1}$

- Unknowns: c_i are rational functions of external kinematic invariants

A Second Motivating Example: High Energy Physics

Example

```
> ring R = (0,p11,p12,p22,e34,m1,m2,m3), (x1,x2,x3,x4), dp;
> poly D1 = 2*x3*x4*e34+x1*(p11*x1+p12*x2)
.            +(x1*p12+p22*x2)*x2-m1;
> poly D2 = -m2+2*x3*x4*e34-2*p11*x1+p11+x1*(p11*x1+p12*x2)
.            +(x1*p12+p22*x2)*x2-2*p12*x2;
> poly D3 = 2*x3*x4*e34+2*x1*p12-m3+x1*(p11*x1+p12*x2)+p22
.            +(x1*p12+p22*x2)*x2+2*p22*x2;
> ideal I = D1, D2, D3;
> ideal GI = groebner(I);
```

Joint project with Pierpaolo Mastrolia, Tiziano Peraro, and Janko Böhm.

First Challenge: Efficiency of Fundamental Algorithms

First Challenge: Efficiency of Fundamental Algorithms

Consider key algorithms in SINGULAR:

Consider key algorithms in SINGULAR:

Fundamental stuff

- Gröbner and standard Bases;

First Challenge: Efficiency of Fundamental Algorithms

Consider key algorithms in SINGULAR:

Fundamental stuff

- Gröbner and standard Bases;
- syzygies and free resolutions;

First Challenge: Efficiency of Fundamental Algorithms

Consider key algorithms in SINGULAR:

Fundamental stuff

- Gröbner and standard Bases;
- syzygies and free resolutions;
- polynomial factorization.

First Challenge: Efficiency of Fundamental Algorithms

Consider key algorithms in SINGULAR:

Fundamental stuff

- Gröbner and standard Bases;
- syzygies and free resolutions;
- polynomial factorization.

Higher level stuff

- Primary decomposition; algorithms of Gianni-Trager-Zacharias, Shimoyama-Yokoyama, Eisenbud-Huneke-Vasconcelos: `primdec.lib`.;

First Challenge: Efficiency of Fundamental Algorithms

Consider key algorithms in SINGULAR:

Fundamental stuff

- Gröbner and standard Bases;
- syzygies and free resolutions;
- polynomial factorization.

Higher level stuff

- Primary decomposition; algorithms of Gianni-Trager-Zacharias, Shimoyama-Yokoyama, Eisenbud-Huneke-Vasconcelos: `primdec.lib`.;
- normalization: algorithms of de Jong, Greuel-Laplagne-Seelisch.

First Challenge: Efficiency of Fundamental Algorithms

Consider key algorithms in SINGULAR:

Fundamental stuff

- Gröbner and standard Bases;
- syzygies and free resolutions;
- polynomial factorization.

Higher level stuff

- Primary decomposition; algorithms of Gianni-Trager-Zacharias, Shimoyama-Yokoyama, Eisenbud-Huneke-Vasconcelos: `primdec.lib`.;
- normalization: algorithms of de Jong, Greuel-Laplagne-Seelisch.
- means to analyse singularities: Hamburger-Noether expansions, blow-ups, resolution of singularities, and more.

Example

- Dereje Kifle Boku, Wolfram Decker, Claus Fieker, Andreas Steenpass: *Gröbner Bases over Algebraic Number Fields.* Proceedings of the 2015 International Workshop on Parallel Symbolic Computation. PASCO 15. Bath, United Kingdom: ACM, 2015, 1624.

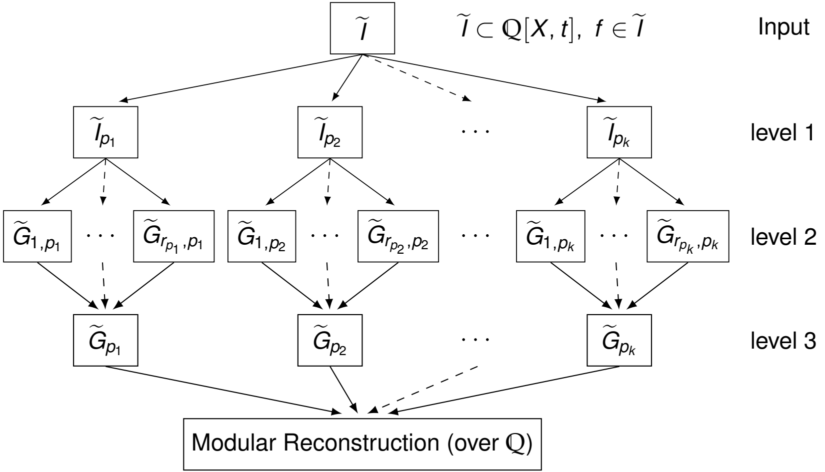
First Challenge: Efficiency of Fundamental Algorithms

Example

- Dereje Kifle Boku, Wolfram Decker, Claus Fieker, Andreas Steenpass: *Gröbner Bases over Algebraic Number Fields.* Proceedings of the 2015 International Workshop on Parallel Symbolic Computation. PASCO 15. Bath, United Kingdom: ACM, 2015, 1624.
- Burcin Eröcal, Oleksandr Motsak, Frank-Olaf Schreyer, Andreas Steenpass: *Refined Algorithms to Compute Syzygies.* J. Symb. Comput. 74 (2016), 308327.

Ongoing work: Gröbner bases over rational function fields, various approaches to computing syzygies.

Example: Gröbner Bases over Number Fields



Second Challenge: Parallelization

Parallelization is a fundamental challenge to all CAS from both a computer science and a mathematical point of view.

Second Challenge: Parallelization

Parallelization is a fundamental challenge to all CAS from both a computer science and a mathematical point of view.

Computer science point of view

In principle, there are two types of parallelization:

- **Coarse-grained parallelisation** works by starting different processes not sharing memory space, and elaborate but infrequent ways of exchanging global data.

Second Challenge: Parallelization

Parallelization is a fundamental challenge to all CAS from both a computer science and a mathematical point of view.

Computer science point of view

In principle, there are two types of parallelization:

- **Coarse-grained parallelisation** works by starting different processes not sharing memory space, and elaborate but infrequent ways of exchanging global data.
- **Fine-grained parallelisation** works with multiple threads in a single process sharing both memory space and global data, and typically with frequent but efficient communications.

Second Challenge: Parallelization

Parallelization is a fundamental challenge to all CAS from both a computer science and a mathematical point of view.

Computer science point of view

In principle, there are two types of parallelization:

- **Coarse-grained parallelisation** works by starting different processes not sharing memory space, and elaborate but infrequent ways of exchanging global data.
- **Fine-grained parallelisation** works with multiple threads in a single process sharing both memory space and global data, and typically with frequent but efficient communications.

The latter requires, for example, to make the memory management of the CAS thread-safe in an effective way.

The Four Cornerstones

GAP: group and representation theory, general purpose high level interpreted programming language.

SINGULAR: polynomial computations in algebraic geometry, commutative algebra, and singularity theory.

polymake: convex polytopes, polyhedral fans, simplicial complexes, related objects from combinatorics & geometry.

ANTIC: number theoretic software, computations in and with number fields and generic finitely presented rings.

Example

```
> LIB("parallel.lib","random.lib");
> ring R = 0,x(1..4),dp;
> ideal I = randomid(maxideal(3),3,100);
```

Example

```
> LIB("parallel.lib","random.lib");
> ring R = 0,x(1..4),dp;
> ideal I = randomid(maxideal(3),3,100);
> proc sizeStd(ideal I, string monord){
      def R = basering; list RL = ringlist(R);
      RL[3][1][1] = monord; def S = ring(RL); setring(S);
      return(size(std(imap(R,I))));}
```

Example

```
> LIB("parallel.lib","random.lib");
> ring R = 0,x(1..4),dp;
> ideal I = randomid(maxideal(3),3,100);
> proc sizeStd(ideal I, string monord){
      def R = basering; list RL = ringlist(R);
      RL[3][1][1] = monord; def S = ring(RL); setring(S);
      return(size(std(imap(R,I))));}
> list commands = "sizeStd","sizeStd";
> list args = list(I,"lp"),list(I,"dp");
```

Example

```
> LIB("parallel.lib","random.lib");
> ring R = 0,x(1..4),dp;
> ideal I = randomid(maxideal(3),3,100);
> proc sizeStd(ideal I, string monord){
      def R = basering; list RL = ringlist(R);
      RL[3][1][1] = monord; def S = ring(RL); setring(S);
      return(size(std(imap(R,I))));}
> list commands = "sizeStd","sizeStd";
> list args = list(I,"lp"),list(I,"dp");
> parallelWaitFirst(commands, args);
  [1] empty list
  [2] 11
```

Coarse Grained Parallelism in SINGULAR by Example

Example

```
> LIB("parallel.lib","random.lib");
> ring R = 0,x(1..4),dp;
> ideal I = randomid(maxideal(3),3,100);
> proc sizeStd(ideal I, string monord){
      def R = basering; list RL = ringlist(R);
      RL[3][1][1] = monord; def S = ring(RL); setring(S);
      return(size(std(imap(R,I))));}
> list commands = "sizeStd","sizeStd";
> list args = list(I,"lp"),list(I,"dp");
> parallelWaitFirst(commands, args);
  [1] empty list
  [2] 11
> parallelWaitAll(commands, args);
  [1] 55
  [2] 11
```

Second Challenge: Parallelization

Mathematical point of view

- There are many algorithms whose basic strategy is sequential in nature. The systematic design of parallel algorithms in areas where no such algorithms exist is a tremendous task. For computations over the rationals, it is important to identify algorithms which allow parallelization via modular methods. Here, mathematical ideas are needed to design the final verification steps.

Second Challenge: Parallelization

Mathematical point of view

- There are many algorithms whose basic strategy is sequential in nature. The systematic design of parallel algorithms in areas where no such algorithms exist is a tremendous task. For computations over the rationals, it is important to identify algorithms which allow parallelization via modular methods. Here, mathematical ideas are needed to design the final verification steps.
- A prominent example of an approach which is inherently parallel is Villamayor's constructive version of Hironaka's desingularization theorem.

Example: Resolution of Singularities

Theorem (Hironaka, 1964)

For every algebraic variety over a field K with $\operatorname{char} K = 0$ *a desingularization can be obtained by a finite sequence of blow-ups along smooth centers.*

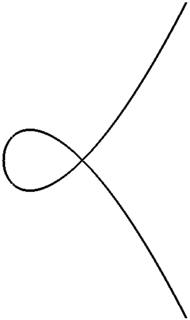
Example: Resolution of Singularities

Theorem (Hironaka, 1964)

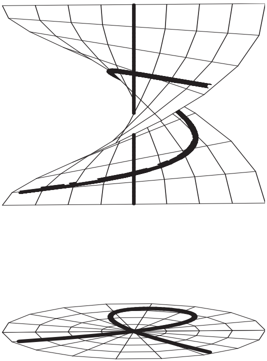
For every algebraic variety over a field K with $\operatorname{char} K = 0$ a desingularization can be obtained by a finite sequence of blow-ups along smooth centers.

For example, resolve the

node



by one blow-up



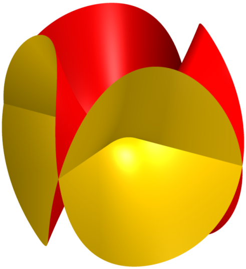
which replaces the singular point inside the plane by a line, hence separating the two branches of the curve intersecting in the singularity.

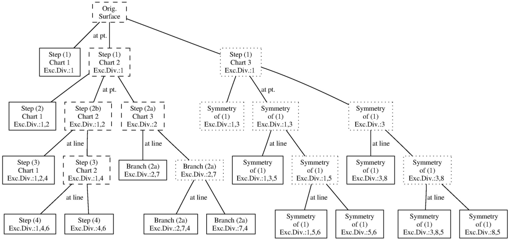
Example: Resolution of Singularities

Working with blow-ups means to work with different charts.

Example: Resolution of Singularities

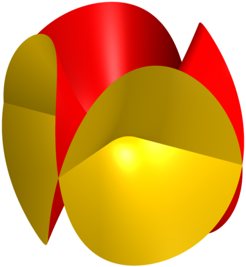
Working with blow-ups means to work with different charts.
In this way, the resolution of singularities leads to a tree of charts. Here is the graph for resolving the singularities of $z^2 - x^2y^2 = 0$

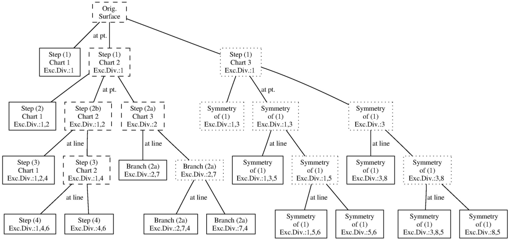




Example: Resolution of Singularities

Working with blow-ups means to work with different charts.
In this way, the resolution of singularities leads to a tree of charts. Here is the graph for resolving the singularities of $z^2 - x^2y^2 = 0$

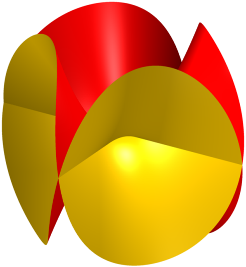


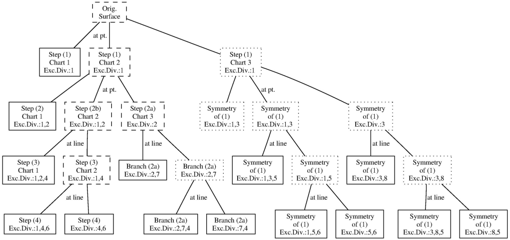


The implementation of a massively parallel general framework for Hironaka-type resolution algorithms may help to find evidence on the resolution of singularities in characteristic p.

Example: Resolution of Singularities

Working with blow-ups means to work with different charts.
In this way, the resolution of singularities leads to a tree of charts. Here is the graph for resolving the singularities of $z^2 - x^2y^2 = 0$





The implementation of a massively parallel general framework for Hironaka-type resolution algorithms may help to find evidence on the resolution of singularities in characteristic p. Use the GPI-Space framework by Fraunhofer ITWM at Kaiserslautern.

Third Challenge: Make More and More of the Abstract Concepts of Pure Mathematics Constructive

Example: Cohomology

Typical applications of Gröbner Bases and Syzygies in the non-commutative case:

Example: Cohomology

Typical applications of Gröbner Bases and Syzygies in the non-commutative case:

Example (De Rham cohomology)

Use the Weyl algebra to compute the de Rham cohomology of complements of affine varieties. Algorithm by Uli Walther, implemented by Cornelia Rottner in SINGULAR.

Example: Cohomology

Typical applications of Gröbner Bases and Syzygies in the non-commutative case:

Example (De Rham cohomology)

Use the Weyl algebra to compute the de Rham cohomology of complements of affine varieties. Algorithm by Uli Walther, implemented by Cornelia Rottner in SINGULAR.

Example (Sheaf cohomology)

Use the exterior algebra to compute the cohomology of coherent sheaves on projective space via the constructive version of the Bernstein-Gel'fand-Gel'fand (BGG) correspondence by Eisenbud-Fløystad-Schreyer.

Notation

Let V be a vector space of dimension $n+1$ over a field K with dual space $W = V^*$, and with fixed dual bases $e_0, \ldots, e_n$ and $x_0, \ldots, x_n$.

The BGG Correspondence

Notation

Let V be a vector space of dimension $n+1$ over a field K with dual space $W = V^*$, and with fixed dual bases $e_0, \ldots, e_n$ and $x_0, \ldots, x_n$. Let $S = \mathrm{Sym}_K(W) = K[x_0, \ldots, x_n]$, and let $E = \bigwedge V$.

Notation

Let V be a vector space of dimension $n+1$ over a field K with dual space $W = V^*$, and with fixed dual bases $e_0, \ldots, e_n$ and $x_0, \ldots, x_n$. Let $S = \mathrm{Sym}_K(W) = K[x_0, \ldots, x_n]$, and let $E = \bigwedge V$. We grade S and E by taking elements of W to have degree 1, and elements of V to have degree -1.

The BGG Correspondence

Notation

Let V be a vector space of dimension $n+1$ over a field K with dual space $W = V^*$, and with fixed dual bases $e_0, \dots, e_n$ and $x_0, \dots, x_n$. Let $S = \mathrm{Sym}_K(W) = K[x_0, \dots, x_n]$, and let $E = \bigwedge V$. We grade S and E by taking elements of W to have degree 1, and elements of V to have degree -1. Let $\mathbb{P}^n = \mathbb{P}(V)$ be the projective space of lines in V, so that S is the homogeneous coordinate ring of $\mathbb{P}^n$.

Notation

Let V be a vector space of dimension $n+1$ over a field K with dual space $W = V^*$, and with fixed dual bases $e_0, \dots, e_n$ and $x_0, \dots, x_n$. Let $S = \mathrm{Sym}_K(W) = K[x_0, \dots, x_n]$, and let $E = \bigwedge V$. We grade S and E by taking elements of W to have degree 1, and elements of V to have degree -1. Let $\mathbb{P}^n = \mathbb{P}(V)$ be the projective space of lines in V, so that S is the homogeneous coordinate ring of $\mathbb{P}^n$.

The BGG correspondence relates bounded complexes of coherent sheaves on $\mathbb{P}^n$ and minimal doubly infinite free resolutions over E.

The BGG Correspondence

Notation

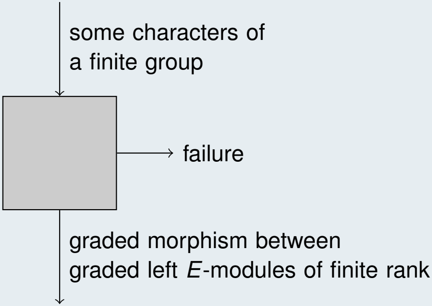
Let V be a vector space of dimension $n+1$ over a field K with dual space $W = V^*$, and with fixed dual bases $e_0, \ldots, e_n$ and $x_0, \ldots, x_n$. Let $S = \mathrm{Sym}_K(W) = K[x_0, \ldots, x_n]$, and let $E = \bigwedge V$. We grade S and E by taking elements of W to have degree 1, and elements of V to have degree -1. Let $\mathbb{P}^n = \mathbb{P}(V)$ be the projective space of lines in V, so that S is the homogeneous coordinate ring of $\mathbb{P}^n$.

The BGG correspondence relates bounded complexes of coherent sheaves on $\mathbb{P}^n$ and minimal doubly infinite free resolutions over E. In particular, it associates to each finitely generated graded S-module M a so-called *Tate resolution* which only depends on the sheafification $\widetilde{M}$ and which “reflects” the cohomology of $\widetilde{M}$ and all its twists.

A `homalg` Example: From Groups to Vector Bundles

A homalg Example: From Groups to Vector Bundles

Mohamed Barakat’s dream to construct low rank vecor bundles on projective space using homalg



A homalg Example: From Groups to Vector Bundles

```
gap> LoadPackage( "repsn" );;
gap> LoadPackage( "GradedModules" );;
gap> G := SmallGroup( 1000, 93 );
<pc group of size 1000 with 6 generators>
gap> Display( StructureDescription( G ) );
((C5 x C5) : C5) : Q8

gap> V := Irr( G )[6];; Degree( V );
5
gap> T0 := Irr( G )[5];; Degree( T0 );
2
gap> T1 := Irr( G )[8];; Degree( T1 );
5
gap> mu0 := ConstructTateMap( V, T0, T1, 2 );
<A homomorphism of graded left modules>
```

A homalg Example:: From groups to Vector Bundles

```
gap> A := HomalgRing( mu0 );
Q{e0,e1,e2,e3,e4}
(weights: [ -1, -1, -1, -1, -1 ])
gap> M:=GuessModuleOfGlobalSectionsFromATateMap(2, mu0);;
gap> ByASmallerPresentation( M );
<A graded non-zero module presented by 92
  relations for 19 generators>

gap> S := HomalgRing( M );
Q[x0,x1,x2,x3,x4]
(weights: [ 1, 1, 1, 1, 1 ])
gap> ChernPolynomial( M );
( 2 | 1-h+4*h^2 ) -> P^4
gap> tate := TateResolution( M, -5, 5 );;
```

```
gap> Display( BettiTable( tate ) );
total:   100  37  14  10   5   2   5  10  14  37 100   ?   ?   ?   ?
----------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
    4:   100  35   4   .   .   .   .   .   .   .   .   0   0   0   0
    3:     *   .   2  10  10   5   .   .   .   .   .   .   0   0   0
    2:     *   *   .   .   .   .   .   2   .   .   .   .   .   0   0
    1:     *   *   *   .   .   .   .   .   .   5  10  10   2   .   0
    0:     *   *   *   *   .   .   .   .   .   .   .   .   4  35 100
----------|---|---|---|---|---|---|---|---|---|---|---|---|---|---S
twist:    -9  -8  -7  -6  -5  -4  -3  -2  -1   0   1   2   3   4   5
---------------------------------------------------------------------
Euler:   100  35   2 -10 -10  -5   0   2   0  -5 -10 -10   2  35 100
```

Third Challenge: Make More and More of the Abstract Concepts of Pure Mathematics Constructive

Exploit derived equivalences in computer algebra

The abstract language of derived categories provides a unifying and refining framework for constructions of homological algebra, duality, and cohomology theories in algebraic geometry.

Third Challenge: Make More and More of the Abstract Concepts of Pure Mathematics Constructive

Exploit derived equivalences in computer algebra

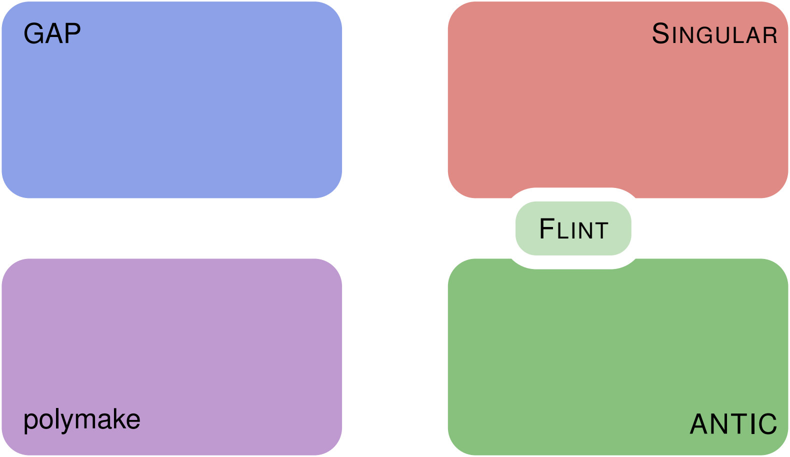
The abstract language of derived categories provides a unifying and refining framework for constructions of homological algebra, duality, and cohomology theories in algebraic geometry. Modeling such concepts in computer algebra is a fundamental task for the years to come.

Third Challenge: Make More and More of the Abstract Concepts of Pure Mathematics Constructive

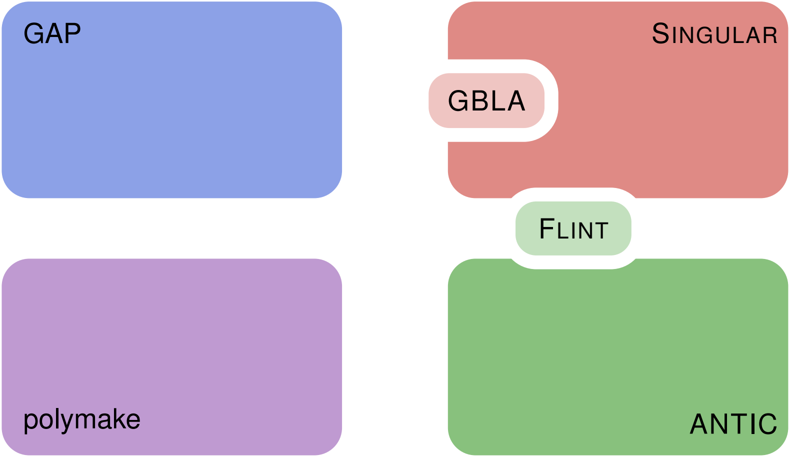
Exploit derived equivalences in computer algebra

The abstract language of derived categories provides a unifying and refining framework for constructions of homological algebra, duality, and cohomology theories in algebraic geometry. Modeling such concepts in computer algebra is a fundamental task for the years to come. In fact, the relationship between computer algebra and higher mathematical structures is of mutual benefit. Derived equivalences can, for example, be utilised to translate problems into an entirely different context with more efficient data structures and reduced complexity.

The Cornerstone Systems Rely on Packages



The Cornerstone Systems Rely on Packages



Software for Number Theory

- FLINT (Bill Hart, Fredrik Johansson, et. al.) provides specific *highly optimized* implementations in C of arithmetic with numbers, polynomials, power series and matrices over many base rings;
- ANTIC (Claus Fieker, Bill Hart, Tommy Hofmann): Fastest known library for number field arithmetic; written in a mixture of C and JULIA;

Software for Number Theory

- FLINT (Bill Hart, Fredrik Johansson, et. al.) provides specific *highly optimized* implementations in C of arithmetic with numbers, polynomials, power series and matrices over many base rings;
- ANTIC (Claus Fieker, Bill Hart, Tommy Hofmann): Fastest known library for number field arithmetic; written in a mixture of C and JULIA;
- NEMO (Bill Hart, Tommy Hofmann, Fredrik Johansson, Oleksandr Motsak): Implementation of recursive, generic rings in JULIA;
- HECKE (Claus Fieker, Tommy Hofmann): Class groups and much more; written in JULIA.

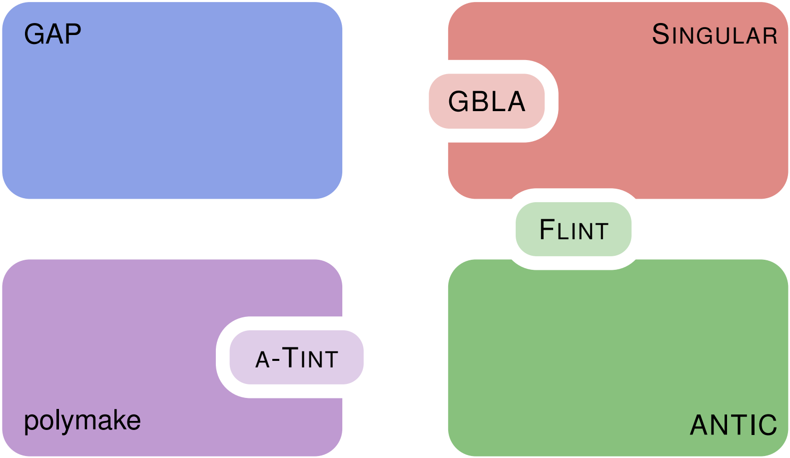
Resultant benchmark: NEMO

```
R = GF(17^11)
S = R[y]
T = S/(y^3 + 3x*y + 1)
U = T[z]
f = T(3y^2 + y + x)*z^2 + T((x + 2)*y^2 + x + 1)*z + T(4x*y + 3)
g = T(7y^2 - y + 2x + 7)*z^2 + T(3y^2 + 4x + 1)*z + T((2x + 1)*y + 1)
s = f^12
t = (s + g)^12
time r = resultant(s, t)
```

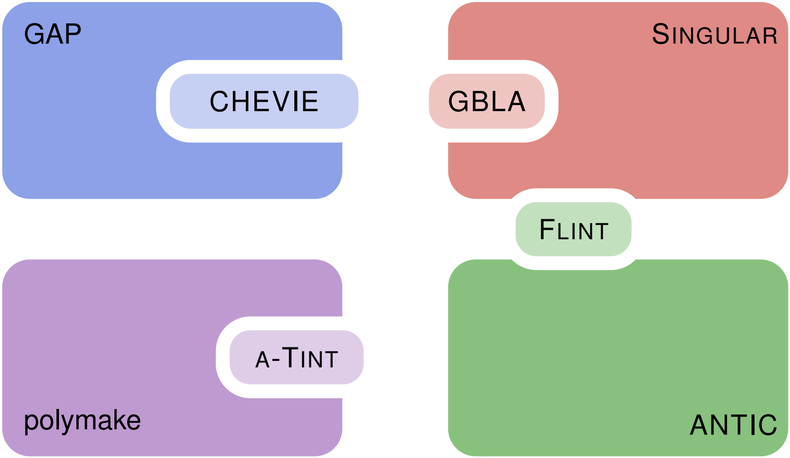
This benchmark is designed to test generics and computation of the resultant.

SageMath 6.8	Magma V2.21-4	Nemo-0.3
179907s	82s	2s

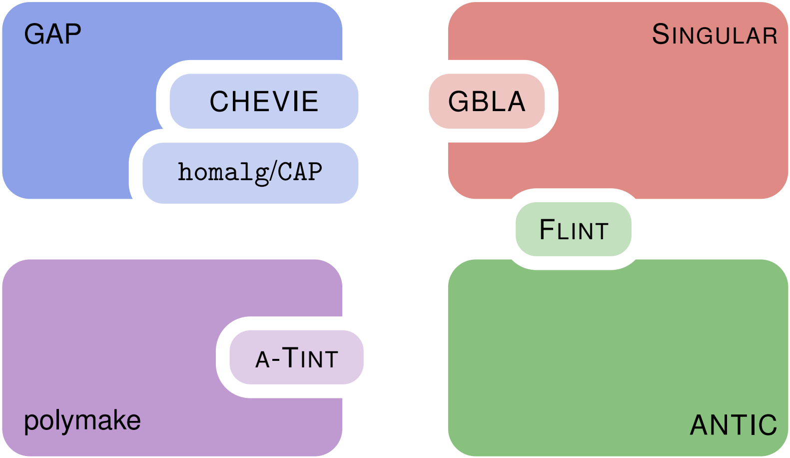
Other Packages Rely on the Cornerstones



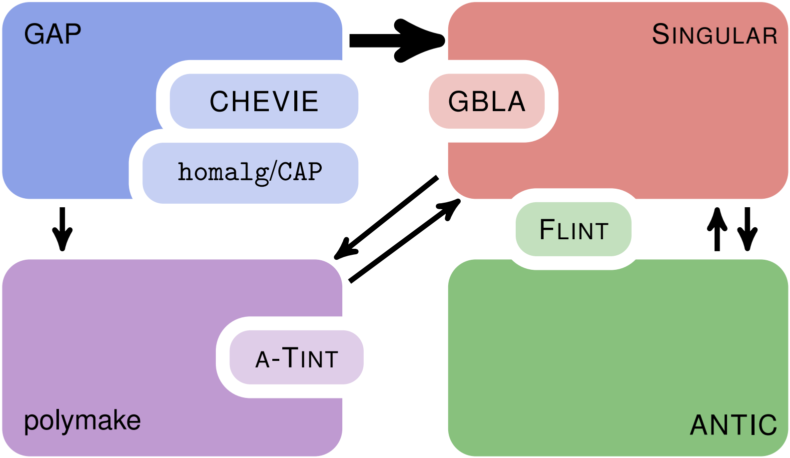
Other Packages Rely on the Cornerstones



Other Packages Rely on the Cornerstones

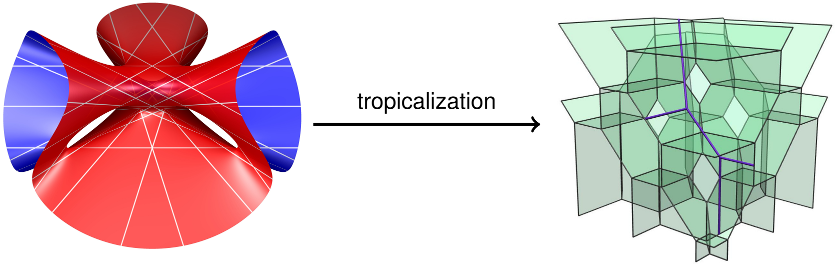


Other Packages Rely on the Cornerstones



What is tropical geometry?

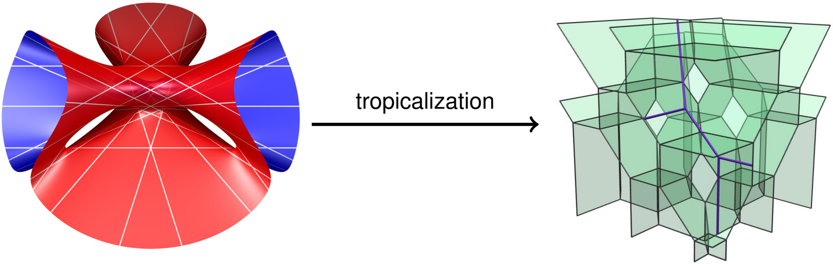
- Tropical geometry is a piece-vise linear version of algebraic geometry.
- Algebraic objects become discrete / polyhedral objects. Tropical varieties are polyhedral complexes.



Software for Tropical Geometry

What is tropical geometry?

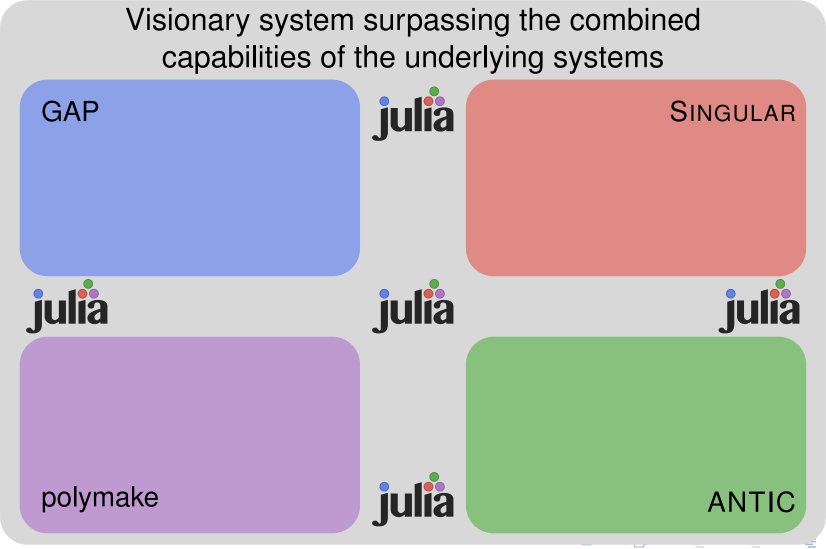
- Tropical geometry is a piece-vise linear version of algebraic geometry.
- Algebraic objects become discrete / polyhedral objects. Tropical varieties are polyhedral complexes.



Some Software

GFAN (Anders Jensen), SINGULAR (Yue Ren, Thomas Markwig), A-TINT (Simon Hampe)

Fourth Challenge: Integration and Interaction of the Computer Algebra Systems and Libraries Involved



Fourth Challenge: Integration and Interaction of the Computer Algebra Systems and Libraries Involved

Visionary system surpassing the combined capabilities of the underlying systems

GAP

julia

SINGULAR

... we can much easier implement:

- Multigraded equivariant Cox rings of toric varieties over number fields

julia

julia

polymake

julia

ANTIC

Fourth Challenge: Integration and Interaction of the Computer Algebra Systems and Libraries Involved

Visionary system surpassing the combined capabilities of the underlying systems

GAP

julia

SINGULAR

... we can much easier implement:

- Multigraded equivariant Cox rings of toric varieties over number fields
- Graphs of groups in division algebras

julia

julia

polymake

julia

ANTIC

Fourth Challenge: Integration and Interaction of the Computer Algebra Systems and Libraries Involved

Visionary system surpassing the combined capabilities of the underlying systems

GAP

julia

SINGULAR

... we can much easier implement:

- Multigraded equivariant Cox rings of toric varieties over number fields
- Graphs of groups in division algebras
- Matrix groups over a polynomial ring over a number field

julia

julia

polymake

julia

ANTIC

Fourth Challenge: Integration and Interaction of the Computer Algebra Systems and Libraries Involved

Visionary system surpassing the combined capabilities of the underlying systems

GAP

julia

SINGULAR

... we can much easier implement:

- Multigraded equivariant Cox rings of toric varieties over number fields
- Graphs of groups in division algebras
- Matrix groups over a polynomial ring over a number field
- Gröbner complexes over fields with a discrete valuation

julia

julia

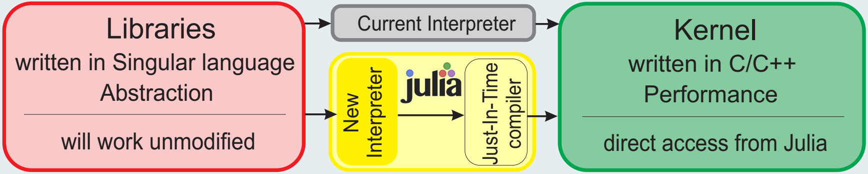
polymake

julia

ANTIC

Fifth Challenge: A Convenient Hierarchy of Languages

System design of SINGULAR



Sixth Challenge: Create and Integrate Data Bases Relevant to Research

For research collaboration within the mathematical community it is essential to collect results of extensive and often time-consuming calculations in databases and make them accessible for further research.

Sixth Challenge: Create and Integrate Data Bases Relevant to Research

For research collaboration within the mathematical community it is essential to collect results of extensive and often time-consuming calculations in databases and make them accessible for further research. Extremely useful examples are collections of classes of mathematical objects such as the SmallGroups Library, which is distributed as a GAP package, or the Graded Ring Database by Gavin Brown and Alexander Kasprzyk with coauthors (MAGMA, PALP, POLYMAKE, and LATTE).

Sixth Challenge: Create and Integrate Data Bases Relevant to Research

For research collaboration within the mathematical community it is essential to collect results of extensive and often time-consuming calculations in databases and make them accessible for further research. Extremely useful examples are collections of classes of mathematical objects such as the SmallGroups Library, which is distributed as a GAP package, or the Graded Ring Database by Gavin Brown and Alexander Kasprzyk with coauthors (MAGMA, PALP, POLYMAKE, and LATTE).

Developing generic concepts to make mathematical data easy to access by the mathematical community, providing both a human interface for browsing and a computer interface to feed the data directly into the computer algebra systems for further studies, is another challenge.

Seventh Challenge: Easy Access

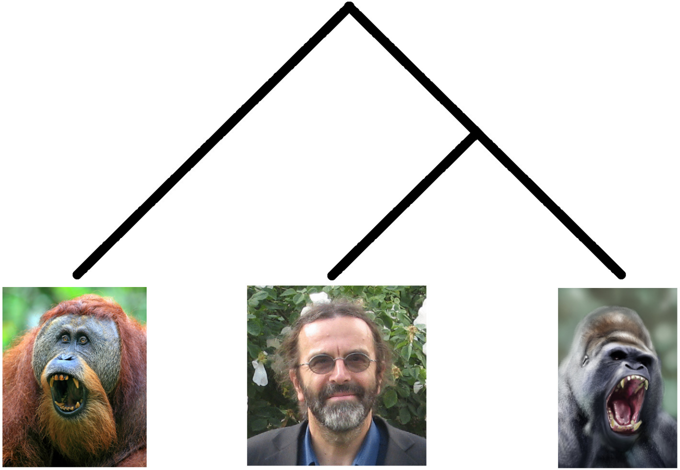
The computational methods provided by our open source computer algebra systems support powerful applications to areas far beyond pure mathematics, for example to computational biology, algebraic vision, and physics.

Seventh Challenge: Easy Access

The computational methods provided by our open source computer algebra systems support powerful applications to areas far beyond pure mathematics, for example to computational biology, algebraic vision, and physics. However, the broad scope and generality of the systems is rarely needed by scientists working in these areas, and may even pose an obstacle.

An Example From Phylogenetics

Mathematical Models of DNA Mutation



Example

The strand symmetric model reflects the symmetry of the double-stranded structure of DNA.

Example

The strand symmetric model reflects the symmetry of the double-stranded structure of DNA. Computing the invariants of such a model on the three loaf claw tree was an open problem for some time.

An Example From Phylogenetics

Example

The strand symmetric model reflects the symmetry of the double-stranded structure of DNA. Computing the invariants of such a model on the three loaf claw tree was an open problem for some time. A direct computation by Fabian Engelmann in SINGULAR using Fourier coordinates and a Hilbert driven computation of the Gröbner basis confirmed the result.

Example

The strand symmetric model reflects the symmetry of the double-stranded structure of DNA. Computing the invariants of such a model on the three loaf claw tree was an open problem for some time. A direct computation by Fabian Engelmann in SINGULAR using Fourier coordinates and a Hilbert driven computation of the Gröbner basis confirmed the result. The computation took 26 CPU days, the resulting Gröbner basis has 416812 elements.

An Example From Phylogenetics

Example

The strand symmetric model reflects the symmetry of the double-stranded structure of DNA. Computing the invariants of such a model on the three loaf claw tree was an open problem for some time. A direct computation by Fabian Engelmann in SINGULAR using Fourier coordinates and a Hilbert driven computation of the Gröbner basis confirmed the result. The computation took 26 CPU days, the resulting Gröbner basis has 416812 elements. The problem was first solved by Long and Sullivant.

Seventh Challenge: Easy Access

The computational methods provided by our open source computer algebra systems support powerful applications to areas far beyond pure mathematics, for example to computational biology, algebraic vision, and physics. However, the broad scope and generality of the systems is rarely needed by scientists working in these areas, and may even pose an obstacle.

Seventh Challenge: Easy Access

The computational methods provided by our open source computer algebra systems support powerful applications to areas far beyond pure mathematics, for example to computational biology, algebraic vision, and physics. However, the broad scope and generality of the systems is rarely needed by scientists working in these areas, and may even pose an obstacle. It is therefore important to provide intuitive user interfaces tailored to specific applications, e.g. customisable graphical notebooks which hide irrelevant features.

Seventh Challenge: Easy Access

The computational methods provided by our open source computer algebra systems support powerful applications to areas far beyond pure mathematics, for example to computational biology, algebraic vision, and physics. However, the broad scope and generality of the systems is rarely needed by scientists working in these areas, and may even pose an obstacle. It is therefore important to provide intuitive user interfaces tailored to specific applications, e.g. customisable graphical notebooks which hide irrelevant features.

The Horizon 2020 European Research Infrastructures project OpenDreamKit centers around this problem.

Seventh Challenge: Easy Access

The computational methods provided by our open source computer algebra systems support powerful applications to areas far beyond pure mathematics, for example to computational biology, algebraic vision, and physics. However, the broad scope and generality of the systems is rarely needed by scientists working in these areas, and may even pose an obstacle. It is therefore important to provide intuitive user interfaces tailored to specific applications, e.g. customisable graphical notebooks which hide irrelevant features.

The Horizon 2020 European Research Infrastructures project OPENDREAMKIT centers around this problem. It aims in particular at providing DOCKER images resp. creating JUPYTER notebooks for the use of the systems.

Seventh Challenge: Easy Access

Problem: Installing several programs with different interfaces can be tricky...

Seventh Challenge: Easy Access

Problem: Installing several programs with different interfaces can be tricky...
Solution: Provide complete enviroments containing all software necessary!

Seventh Challenge: Easy Access

Problem: Installing several programs with different interfaces can be tricky...
Solution: Provide complete enviroments containing all software necessary!

- DOCKER provides software in predefined environments

Seventh Challenge: Easy Access

Problem: Installing several programs with different interfaces can be tricky...
Solution: Provide complete enviroments containing all software necessary!

- DOCKER provides software in predefined environments
- Easy one-command installation

Seventh Challenge: Easy Access

Problem: Installing several programs with different interfaces can be tricky...
Solution: Provide complete enviroments containing all software necessary!

- DOCKER provides software in predefined environments
- Easy one-command installation
- No performance loss

Seventh Challenge: Easy Access

Problem: Installing several programs with different interfaces can be tricky...
Solution: Provide complete enviroments containing all software necessary!

- DOCKER provides software in predefined environments
- Easy one-command installation
- No performance loss
- Even works on Windows

Seventh Challenge: Easy Access

Problem: Installing several programs with different interfaces can be tricky...
Solution: Provide complete enviroments containing all software necessary!

- DOCKER provides software in predefined environments
- Easy one-command installation
- No performance loss
- Even works on Windows

Using DOCKER, starting GAP with connection to POLYMAKE and SINGULAR gets as easy as

```
docker run -it sppcomputeralgebra/sppdocker gap
```

Seventh Challenge: Easy Access

jupyter Demo Last Checkpoint: 4 minutes ago (autosaved)

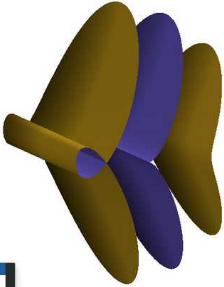
File Edit View Insert Cell Kernel Help | Singular

Code Cell Toolbar: None

```
In [5]: poly logo = ((x+3)^3 + 2*(x+3)^2 - y^2)*(x^3 - y^2)*((x-3)^3-2*(x-3)^2-y^2);

In [6]: LIB "surf_jupyter.lib";
Out[6]: // ** loaded /usr/local/bin/../share/singular/LIB/surf_jupyter.lib (0.0.0.1,Jan_2016)

In [7]: plot_jupyter(logo);
```



```
In [8]: (logo);
In [ ]: st
```

st
status
std
stdfglm
stdhilb
string